Data mining log file streams for the detection of anomalies

Brian Green

A thesis submitted to
Auckland University of Technology
in fulfilment of the requirements for the degree
of
Master of Computer and Information Sciences

4 September 2015

Auckland University of Technology School of Computing and Mathematical Sciences

Contents

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), no material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning."

Brian Green

Acknowledgements

I would like to thank my supervisor Dr. Russel Pears for guiding me on this journey, David Swazbrook and Aaron Cheeseman for their experience and insights in both managing log files and guiding my programing efforts. Roy Cullum for allowing me the time and space to progress this work to completion and finally my biggest supporter who kept me on task when required and for having a keen mind and critical set of eyes when reviewing this thesis, Maryann Green.

This work is dedicated to the memory of Jean Green my mother who passed away half way through this journey of higher education.

List of Abbreviations

- CRISP-DM – Cross Industry Standard Process for Data Mining
- DB – Database
- DHCP – Dynamic Host Configuration Protocol
- EM – Earth Mover's
- EMD – Earth Mover's Distance
- EPS – Events Per Second
- HMAT – "Histogram Matrix" (Frei & Rennhard, 2008)
- KDD – Knowledge Discovery in Databases
- NTP – Network Time Protocol
- R – R: A Language and Environment for Statistical Computing
- SSH – Secure Shell

List of Figures

List of Tables

Abstract

Log files play an important part in the day to day running of many systems and services, allowing administrators and other users to gain insights into operational, performance or even security issues but it is now impractical with the volume of files today to manually examine them.

Existing tools in this space largely work by detecting anomalies from log files that have already been stored or by comparing them against known errors (signatures). By data mining log file streams for the detection of anomalies instead, it will allow administrators to reduce the time required to detect anomalies significantly with no signatures or complex settings needing to be maintained.

This paper presents the experimental work undertaken to define a generic, practical and scalable method for anomaly detection in streaming log files by detecting the change to the mix of log events occurring. This was achieved by following a modified CRISP-DM (Cross Industry Standard Process for Data Mining) methodology enabling a broader more flexible approach to the data mining process.

By taking this approach, a solution was developed that employs common log file features together with a weighted earth mover distance metric. This enabled a framework to be developed that can be broadly applied to many log file types. By setting a simple percentile threshold indicating an acceptable level of change, anomaly detection in streaming log files can be achieved.

Chapter 1 - Introduction

Log files are used across many IT operational areas today to investigate security or operational issues that may have occurred to gain further knowledge about these events. These log files typically are examined after something has been detected and further knowledge needs to be gained about what has occurred.

Timely knowledge on when something has changed in the operation of a system can minimise service interruption or security risk to an organisation. By examining these log files, critical insights can be gained, but this requires knowledge on when and where such an examination should take place.

There are many challenges when working with log files across a number of problem areas, ranging from the distribution of events, the volume of data, the wide range of log files formats, and various transport or storage methods. In addition to these challenges given the large volume of log file information it is usually infeasible to manually examine all the log files in order to analysis them for anomalies.

The problem of log file analysis is of direct interest to the author, as his team has to deal with log file analysis as part of their daily operational tasks, this includes responding to security, operational and system performance issues.

Through the work presented in this thesis in data mining log file streams for the detection of anomalies, a more generic approach to anomaly detection (change detection) is envisaged, not looking for one off events which are much more easily detected by signature or other change detection methodologies, but more when the ‘‘mix’’ of predictor attributes has changed enough from a baseline of log activity. To achieve this, new predictor attributes and change detection methodologies will be investigated. This will allow the administrator of a system or service to narrow down the time and location of a change in the mix of log file events in order to discover the changes / anomalies that have taken place in a system or service, thus allowing for an appropriate and timely response.

This thesis presents a generalised and highly scalable approach for anomaly detection which could be easily adapted as a future piece of work into a full scale production system for the detection and visualisation of changes and anomalies in log files. This is

in contrast to a typical streaming pattern matching technique which looks for a single event, or a pre-determined count limit on a number of single events being monitored.

The problems and challenges that exist when trying to examine log files will be examined in the next section.

1.1 The Problem

It can be considered that each system and service has its own pattern of activity. For example, when an email is passed through a mail relay or other IT service normally it will generate a largely matching set of related log events or multiple lines for each event. When a system error or anomaly occurs, typically a different set of log events may be recorded (Fu, Lou, Wang, & Li, 2009) . While the volume of such activity for each system and service will vary across each day, the mix of system and service log file message types should largely remain the same.

Additionally, in a normal business operational environment there is an added pattern to the activity across the day as people arrive and login; they access files, systems and services, and then logout. This normal user activity typically also introduces another mix to the log messages as well as a change in the rate that events are recorded. In a large environment the sheer volume of log files and events presents issues; many gigabytes of logs could be generated each day from one system alone, and thus manually reviewing log files is not only an inefficient method of review, it is also impractical. There is a desire to find a solution to this problem and one way is by focusing attention on detected anomalies only.

They are many different methodologies for anomaly detection in log files (Chandola, Banerjee, & Kumar, 2009); two popular techniques that have been used in many log file systems are clustering and pattern matching (signature based). Clustering is typically used to make spotting outliers much easier so they can be examined further to see if they are an anomaly but it can be computationally expensive. This is in contrast to pattern matching where a known error condition or security event is typically specified via regular expression matching and then alerted or reported to the administrator of the system or a security team. The problem with signature type

approaches is that the signatures must constantly be updated and that they are only really able to detect known anomalies.

The scale of today's ever increasing IT environments can also present challenges. Each service can generate its own log files and there may be more than one service running on a system or instance which in turn is generating log files as well. This can lead to scale issues due to the sheer volume of log messages generated, thus increasing the difficulty of extracting the required knowledge. This scale issue has been examined by many researchers and it is not uncommon to see statements such as ‘‘log files are usually too large and complex to analyse manually’’ (Taerat, Brandt, Gentile, Wong, & Leangsuksun, 2011, p. 285) and ‘‘data by NIDS (Network Intrusions Detection Systems) systems could be overwhelming and it is not an easy task for network administrators to go through the collected logs’’ (Mohammadjafar Esmaeili & Arwa Almadan, 2011).

One further complication is the problem is that log files are usually textual in nature and differ in context, structure, transport and storage methods between all the various systems and service (Taerat et al., 2011) . All these issues compound to add to further challenges when analysing log files.

More recently with the increasing use of cloud technologies (virtual servers and services) the possibility of having systems and services running across a variety of hosting providers and geographic locations will yet again result in another level of complexity to be added to this already challenging area. Some of these challenges and approaches to log file analysis will be examined further through a literature review presented later.

1.2 Motivation for the Study

For more than 20 years the author has been involved in IT system support, and more recently in operation and security management. Over the author's many years in IT, he has examined many log files in order to get to the root cause of an issue or to understand why something has changed. This gives the author not only a generalised domain understanding of this problem area, but also the motivation to search for a more generalised solution.

The inspiration for this work came when the author was reviewing literature for another paper. At that time the author came across two pieces of research in this field, ‘‘Histogram Matrix: Log File Visualization for Anomaly Detection’’ by Frei & Rennhard, 2008 and ‘‘Baler: deterministic, lossless log message clustering tool’’ by Taerat et al., 2011 and it is from these two papers that a possible solution to one area in this problem was formed.

Many tools and techniques have been used or developed by the author over the years in order to help assist in log file analysis, but they tend to be focused on a single need or a small set of requirements. The large majority of them have used some sort of signature type/pattern matching engine in order to do the required analysis. More recently, full text searching has become much easier across many log file sources via tools like Splunk (“Splunk | IT Search for Log Management, Operations, Security and Compliance,” 2010) and Elasticsearch (“Elastic · Revealing Insights from Data (Formerly Elasticsearch),” n.d.). But knowing when and where to go looking and detecting changes remains a problem.

This is the problem that gave the author the motivation to try and search for a solution in this area and present, via the work in this thesis, a generic approach to this problem. The solution needed to be one that was not reliant on generating signatures for specific events, it needed to have the ability to scale to huge size and also provide flexibility in how the data can be examined and visualised when required.

In the author’s employed role his team has to deal with log file analysis issues as part of daily operational tasks which include responding to security, operational and system performance issues. Given the author’s responsibilities, data for this research was readily available, and permission from the organisation was sought and granted for this use.

Through the operational and security community’s work together on solutions in this space, it is hoped that the work and ideas presented in this thesis can be shared back to these communities so it could be developed on in the future. Any code artefact created will be put on a public code repository so other people may benefit from this aspect of this work as well. Many open source tools were used in generating these

experimental results and it is but a small way of showing the author's appreciation of all the work others have done that have made this research possible.

1.3 Research Contributions

This thesis offers a generic approach based on common log file features that could be widely applied across many types of log files or other streaming text type problem spaces. Through exploring different ways to detect changes in streaming log files, this thesis contributes to knowledge by offering a new method of detecting anomalies in log files along with offering new insights into detecting change in unstructured log files.

The research reported is significant, because from a security view point on log file systems, knowing when things have changed from a baseline may allow the early detection of a compromised system or service. This work offers a generalised approach to achieving this with very little setup cost. Once an anomaly is detected immediate security incident response measures can be put in place.

Then, through further examination of the log files identified by the methods outlined in this thesis, a compromised system or service and the reasons thereof may be able to be determined. Once the method of compromise has been determined, additional appropriate mitigation actions can be taken in a timely manner, and development of new controls may also be able to be established.

From an operational view point, the healthy performance of a system or a service is essential to many areas today. Given the very general nature of this approach, detecting when something has changed that is not already picked up by another monitoring system would offer many advantages in system monitoring areas as well. Once detected, checks could also be introduced via other methods such as signature type approaches even quicker. This effectively offers a safety net via its general approach, so while the proposed method is unable to pick up single event types, any time when something causes a large enough change in the mix of log event types to be alerted, an administrator can be used to investigate further if required through other methods and tools.

In order to achieve the goals outlined above specific research questions have been developed to guide the outcomes of this work.

1.4 Research Questions

To structure this data mining experiment a set of research questions were developed in order to guide each iteration of the data mining process as well as the overall approach to the experiment. It was also intended that the realisation of this work into a practical solution should also guide the outcomes of this thesis. With these goals in mind the following research questions were proposed.

Question One:

What log file attributes will give efficient and accurate anomaly predictors from log files?

Question Two:

Which model or set of models will give the best performance for the detection of anomalies in log files based on these anomaly predictors?

Question Three:

Which is the best method to evaluate the model/s generated for the detection of anomalies in log files?

Question Four:

What constraints are required to ensure that the work can be developed in the future to form the basis of a practical and scalable solution for the detection and visualisation of anomalies in log files?

Question one focuses on how to find which attributes will give accurate anomaly predictors from log files; existing work will be examined in order to build a set of attributes that can be evaluated and tested.

Question two gives focus on how to compare the generated features; various methods will be examined in order to establish which ones work best for this problem area. Performance is to be compared both on effectiveness of anomaly detection and bounded by computational cost if it is overly detrimental to the end goal.

Question three directs the research into how to establish thresholds of difference. How to determine when something is ‘‘too’’ different. These should be easily understandable by the end user in order for them to be interpreted and set appropriately given the nature of each log file, or combination of log files.

Question four sets boundaries or constraints on the tools used and created in order to give others the ability to develop this work further. Question four effectively sets the scope of this research and will be examined further below.

1.5 Research Scope

As noted in research question four above, this work is only intended to build the knowledge and develop design ideas and constraints so that in the future a system for the detection of streaming log file anomalies and visualisation of results can be built.

In order for it to be “pragmatic” it should be able to be constructed from readily available applications that can be configured or easily modified to suit the outcomes of this research around generalised anomaly detection and visualisation.

It has to be able to support the anomaly detection methodologies that are ‘‘performant’’ as defined in question two. In addition to the above the entire solution should be scalable so that it could be used from small scale systems all the way to extremely large ‘‘Big Data’’ type environments.

Finally, it should offer a means of alerting and visualisation of the detected anomalies by the thresholds as developed in research question three.

1.6 Organisation of the Thesis

This thesis is organised in the following chapters. A general review of literature is performed in Chapter two which expands on the key papers which inspired pursuing this research topic and further explores the issues and problem areas through the examination of complementary research, articles and appropriate standards.

The research method used is presented in chapter three in order to ground the methodology used in the experimental data mining sections.

Chapter four has the data mining iterations and the experimental results for each iteration that were discovered based on the research method framework. Each iteration has a ‘‘Business and Data Understanding’’ focus in order to describe the goals of the iteration and current data understanding. Also included is ‘‘Data Preparation and Modelling’’ which covers the data, tools and methods used. An evaluation is presented via an experimental study and a summary relating it back to research questions. A final summary concludes the data mining process, and brings together all the results obtained by reviewing the outcomes against the four research questions. A proposed framework is presented at the end of Chapter four. This includes two practical implementations that could be developed to create a working solution using readily available applications and current research.

Chapter five concludes the research, and Chapter six discusses future work, taking into account the outcomes discovered, and ideas which could be developed further.

Chapter 2 - Literature Review

As described in the motivation for this thesis, two main research articles formed the inspiration for the ideas that will be developed to achieve data mining of log file streams for the detection of anomalies. The methodology employed for this literature review is largely inwards focused. The two papers that inspired this thesis will be examined first to show how they inspired and contributed to the overall outcomes of this experiment. Then the ideas and issues raised in these two papers will be explored through the literature thus building the knowledge required in order to create a pragmatic solution for anomaly detection in log files.

The first paper that will be reviewed is "Histogram Matrix: Log File Visualization for Anomaly Detection" (HMAT) by (Frei & Rennhard, 2008) in this paper they present an approach for detecting and visualising anomalies in log files. From a single predictor attribute (word count) they produce a histogram for each log line every hour. From this they applied a bespoke statistical measure as their anomaly detection method based on the standard deviation of each individual bin's values compared to the same bins over the current day or same slot on a previous week as a comparison / baseline option. Their project also supported a visualisation of the results as represented by a matrix on the screen which could be clicked on to drill down to examine the log lines at the matching time. This approach of limiting the logs to the matching time only facilitated the use of other tools such as clustering etc. which might not have coped with processing the full log file.

By taking this approach they made use of the idea that many systems and services have a pattern to their activity which is shown as a sequence of log file events that take place. This mix of log messages will remain the same regardless of the volume of transactions that occur and thus by normalising the volume of the histogram they can compare the mix (Individual bin values over time) in order to detect anomalies.

Areas for improvement to this approach were identified by Frei & Rennhard, 2008 in future work. One issue identified was the limitation of using only one predictor attribute and thus log files whose word count does not vary by very much might not be practical for usage on anomaly detection. They also raised a desire to include clustering into the solution as well. The final area of improvement they identified was

that they would like a way to define rules to remove ‘‘noisy’’ or harmless messages. This desire to remove ‘‘noisy’’ or harmless messages may indicate an oversensitivity in some log file cases caused by the focus on individual bin changes that they have taken.

Inspired by this research, the author began to form a possible solution to investigate ways to move to a streaming approach to speed up detection by utilising the whole histogram instead of just each single bin in order to reduce the implied oversensitivity to single bin changes. This in turn should remove the need for maintaining rule sets to reduce noise.

The author also recognised the need to address the ‘single predictor attribute’ limitation identified by Frei & Rennhard, 2008 and in this experiment a greater range of predictor attributes will need to be discovered and implemented in order to widen applicability of the solution to a wider range of log file types.

The second paper which helped form the overall approach was ‘‘Baler: deterministic, lossless log message clustering tool’’ by Taerat et al., 2011. In this work they tackle the issues of clustering of extremely large log files, and compared the performance of their solution against other common methods and programs used for this purpose. Many of the clustering tools use a multi pass approach for processing data into clusters. Baler achieves the same equivalent outcome by tokenising and clustering the data based on the token attributes in a single pass.

By utilising a single pass approach to generating the token attributes and clustering, it removes the usual memory usage issues that frequently occur in clustering solutions of this type when dealing with very large log files. As part of the initial clustering algorithm, they split the log file line into various token types ‘‘Alpha-Numeric’’, ‘‘Numbers’’, ‘‘Spaces’’ etc. For the ‘‘Alpha-Numeric’’ matches they further tokenised it based on if it is an English word from a pre-loaded list of words, a number or if not matched, then returning an outcome of an alpha numeric token. In examining the clusters generated, they found that they could further consolidate some of the clusters without over clustering the data as a whole by measuring the difference between the clusters based on a string difference algorithm, Perl’s Algorithm::Diff (McQueen, n.d.), and setting a threshold on which to cluster the patterns further.

From this research, additional predictor attributes can be identified and tested based on their successful use in this work as a method for clustering. By combining this work and the ideas developed from HMAT by Frei & Rennhard, 2008, a new approach to anomaly detection in log files could be developed.

The demand to cluster ever larger log files drove the development of Baler by Taerat et al., 2011 and in general the ability to deal with the ever increasing size and volume of log files etc. is driving efforts in many other research areas.

One of these research areas is in dealing with how to transport, store and process these log files efficiently and effectively. One approach in handling this large scale logging problem is ‘‘Chukwa: A system for reliable large-scale log collection’’ by Rabkin & Katz, 2010. In their work they present a method which supports both reliable and fast methods for transport and storage of log files. The system was also designed so that it could support a number of log file collection methods via local agents using syslog or local file access etc. These local agents then sent their logs to a separate collector agent which then split them to a ‘‘fast path’’ processer for immediate (but not reliable) processing and also wrote them (reliably) to Hadoop HDFS file system every five minutes. By doing this it allowed a MapReduce type operation to be able to carried out over the stored datasets while supporting the immediate needs of some systems and services.

This work can be compared to ‘‘Logjam: A scalable unified log file archiver’’ by Cardo, 2011. Logjam was designed for the collection of log files across a cluster of servers. It works by generating an archive of the log and transferring it to long term storage using a naming standard for easy identification. It is intended to not run all the time, but batch process the log file archive operations on a daily, weekly or monthly basis. Where Chukwa was designed to process (in seconds) and store (every five minutes) the log files as soon as possible, Logjam was designed to automate the archiving and storage of the log files on a much longer time scale (days, or weeks).

The volume of log files is also described in information about real world operational log data rates sourced from the SANS Institute, a well-known security information vendor. The SANS Institute regularly produces whitepapers, and while they are not all peer-reviewed papers, they can offer valuable insight into various real life log file event

rates. In one paper they divide the event rates into EPS Steady state, Peak EPS, and Average Peak EPS. They rate the log file event rates this way because while there is an average event rate that must be handled, sustained peaks and peak event rates must also be processed timely and efficiently (Butler, 2009). The balance between timely log file processing and reliable log file process and storage is an ongoing challenge.

To further complicate the situation log files come in many formats and to date there has been no one accepted standard for them. In fact major government sponsored efforts towards creating such a standard such as CEE the ("Common Event Expression: CEE, A Standard Log Language for Event Interoperability in Electronic Systems," 2014) have even been put on hold recently.

Other approaches have been to build systems which attempt to normalise the log files by parsing them using custom matching patterns for each log file type as employed by commercial log file analysis tools such as "sawmill.net - Sawmill - Universal log file analysis and reporting," 2012 and "HP ArcSight Connectors - HP Enterprise Security," 2012. By normalising the logs it does allow for other data mining approaches to be used but limits the solution to log files that have had a custom parser added for them and thus any new formats would have to have a parser added which increases the support and maintenance burden of the solution. In addition to this there is no common format for the normalisation of the log files either. With efforts such as CEE being terminated and handed off to third parties to continue if possible, it appears that there will not be much progress in this area in the near future.

Possibly the most widely supported logging standard is syslog which originates from the widely used Sendmail application. Given its wide de facto use for logging it was formalised as a standard in RFC5425 (Gerhards, 2009). The standard largely focuses on the transport method of the log file but does also define a basic structure to each log event including, source, priority, hostname, process, timestamp and a free form message.

Although message content is free form, usually there is an underlying structure to the log file events. This underlying structure or mix of log file message features arises from the way log file messages are produced within the applications. In most cases these log

events are created by structured print statements e.g. ‘‘sprintf(message, ‘‘Connection from %s port %d’’, ipaddress,portnumber)’’ (R. Vaarandi, 2003).

By leveraging this underlying structure to log messages, researchers have found that they can create models for clustering and other data mining techniques in order to extract further knowledge from log files and look for outliers and/or anomalies. Early work in this area is exemplified by SLCT (Simple Logfile Clustering Tool) by R. Vaarandi, 2003 and improved in LogHound (Risto Vaarandi, 2004) in which he clustered log files based on these underlying properties.

As the size of log files has grown, clustering logs based on the underlying structure has become increasing difficult and tools like SLCT and LogHound are no longer practical to use. More recent research has worked towards extending the scale of log file clustering tools by developing new methods such as partitioning the log file clusters before producing the full cluster summary (Makanju, Zincir-Heywood, & Milios, 2012). The issue of dealing with the huge size of some log files is also the area that ‘‘Baler’’ (Taerat et al., 2011) addressed (as noted previously).

Log files also have another property that could be exploited for means of change and/or anomaly detection as described by Fu et al., 2009 where they implemented a finite state automaton model trained from the log files. This method works as many systems or services exhibit a flow of programmed or scripted events based on a schedule giving a defined mix and temporal pattern to them. All these log file properties introduce areas that could be utilised for change and/or anomaly detection.

There are a number of approaches that could be used for anomaly detection, one possible related approach is described in ‘‘Anomaly detection: A survey’’ by (Chandola et al., 2009) where they broadly grouped anomaly detection into three types, point, contextual and collective. The problem at hand aligns with the collective approach in which they describe an anomaly as being when a collection of related data instances is anomalous in respect to the entire data set. This aligns with the desire to use a change in the mix of the log messages in order to detect change and/or anomalies.

In ‘‘A Review of Anomaly based Intrusion Detection Systems’’ by V Jyothsna, V V Rama Prasad, & K Munivara Prasad, 2011 they also divided the field of anomaly detection, but this time into two groups, signature based and anomaly based. They then further

defined each approach and methods that could be appropriate for use in each. In this work the problem aligns largely with the anomaly detection and it is anticipated that one or more of these methods of anomaly detection could be employed based on how the data mining experiment progresses and work such as this will be informative when deciding the approach to take. Methods they mention which could be of use are statistical based, operational or threshold metric model, statistical moments or mean and standard deviation model, univariate model, multivariate model, time series model, outlier detection model, or user intention based.

When examining the area of data stream mining and anomaly detection Mohammadjafar Esmaeili & Arwa Almadan, 2011 in their work ‘‘Stream Data Mining and Anomaly Detection’’ they used a sliding window approach in combination with a signature based approach in order to detect anomalies by comparing previously clustered data. As discussed previously, a signature type approach is not desired in this work. Signatures need to be updated regularly, are typically difficult to create and maintain, and are very dependent on individual log file types, this is counter to the generalised approach desired in this work. Mohammadjafar Esmaeili & Arwa Almadan's, 2011 implementation of a sliding window may offer alternative methods for change and/or anomaly detection and thus may be useful to consider as part of the overall solution.

Looking further at mining streams of data, in Kholghi, Hassanzadeh, & Keyvanpour, 2010 ‘‘Classification and evaluation of data mining techniques for data stream requirements’’ they classified various approaches for data stream mining. They first grouped each solution based on whether the data required pre-processing or not, and then they grouped each one further based on if it employed clustering, classification, or frequency/time series analysis. Further to this they also classified each solution based on whether it stored the data or not. In this data mining experiment the author does not intend to cluster the data. However, time series analysis, data pre-processing (or not) and data storage are aspects which will be considered in the overall solution

By not clustering the data it is hoped to avoid common scalability limits highlighted by Taerat et al., 2011 such as memory usage and computation time. In addition to this it is hoped that not clustering the data will also reduce any assumptions around the structure of log file information, such as English words or message structure as used in

Baler and highlighted in other approaches identified in the literature review in Chapter 2. Not clustering the data should allow for the broad applicability of this streaming log file based anomaly detection system by minimizing the number of assumptions on the structure of the data..

The data mining experiment will take into account these broad classifications for change and/or anomaly detection and as the data mining process progresses these classifications will help inform decision making and possible approaches.

A more recent approach to processing large data sets was achieved by using MapReduce type operations. Liu, Pan, Cao, & Qiao, 2010 in their work “System Anomaly Detection in Distributed Systems through MapReduce-Based Log Analysis” used MapReduce as the last stage of their anomaly detection system in order to analyse the log file clusters that have been created. By using a MapReduce methodology they greatly improve the overall efficiency of their solution. As part of their work they also produced a visualisation of their results. Using elements of their successful processing and visualisation of large data sets, this experiment will aim to offer a scalable solution.

Finally visualisation of data is another technique that can be used to leverage the knowledge gained via data mining methods on large data sets in order to help guide the user or let them discover new knowledge or information. In fact visualisation is a common technique employed as a way to deal with very large data sets. Ben Shneiderman a seminal author in the field of visualisation presented a visualization mantra in which he states ‘‘Overview First, Zoom and Filter, Details-on-Demand‘‘(Shneiderman, 1996). This ‘‘mantra’’ will guide the final outcome of this experiment, as it will help define the requirements for the production of a practical visualisation solution for this work allowing the user to navigate to the detected change and/or anomaly as required.

Chapter 3 - Research Methodology

3.1 Methodological Alternatives

There are a variety of methodologies that could be used for this data mining experiment, and several different methodologies were examined in order to find one that best suited the desired outcomes of this project. One structured approach to data mining was presented in KDD (Knowledge Discovery in Databases) by Fayyad, Piatetsky-shapiro, & Smyth, 1996. Within this work they described a nine step process, presenting a more detailed process oriented description to the overall activity of knowledge discovery from data.

In more recent efforts to further standardising the approach to data mining, one of the most popular approaches is CRISP-DM (Cross Industry Standard Process for Data Mining). The CRISP-DM standardisation effort was led by five companies from a variety of industries. Standardising the variety of data mining processes was the main driver for this piece of work so that it was ‘‘independent of both industry sector and technology used’’ and that data mining processes are ‘‘less costly, more reliable, more repeatable, more manageable and faster’’ (Wirth & Hipp, 2000)

In ‘‘A survey of Knowledge Discovery and Data Mining process models’’ (Kurgan & Musilek, 2006) they noted that CRISP-DM has been used across a number of projects within research and industry. In addition to this, an online survey in 2007 on the KDnuggets website, an online community for data miners run by Gregory Piatetsky-Shapiro, a leading expert in the field of data mining, and also one of the co-authors of the original KDD paper above, found that CRISP-DM is still the top methodology used for data mining projects. This was further reinforced while this experiment was being undertaken when the same survey was run again in 2014 and the results showed it is still one of the most popular methodologies (Piatetsky, 2014).

There have been several efforts to improve the CRISP-DM methodology, as highlighted by Sharma & Osei-Bryson, 2010 in which they examined a number of data mining methodologies and identified several areas of potential improvement. One area of improvement identified could be achieved by integrating all the various models together to address any weaknesses in any individual one. From this work they

developed a new data mining process model ‘‘Integrated Knowledge Discovery and Data Mining’’ (IKDDM).

Sharma & Osei-Bryson, 2010, p. 11347 also described the original KDDM process as only a checklist type approach, this is in comparison with their work in IKDDM in which the process model they show was very much more prescriptive. In order to evaluate the new data mining process an analytical comparison between IKDDM and CRISP-DM was carried out and they presented the conclusion that IKDDM offered improvements in both effectiveness and efficiency (Sharma, Osei-Bryson, & Kasper, 2012).

The KDDM checklist approach, and the much more prescriptive IKDDM approach can be contrasted with the KDnuggets data mining survey’s second most popular methodology identified as ‘‘My Own.’’ ‘‘My Own’’ is essentially any personal approach that the data miner has chosen to use. In the 2014 KDnuggets survey ‘‘My Own’’ was again identified as the second most popular data mining methodology with 27.5% of the vote, this shows increased preference for customised approaches when compared with the 2007 survey result of 19%.The strong support for ‘‘My Own’’ methodology may indicated that even with a standardised method that some room for personal customisation and flexibility may be required given that data mining approaches, requirements and challenges are constantly changing.

While work on CRISP-DM now appears to be stalled, Piatetsky, 2014 notes that it ‘‘*does not seem to be maintained and adapted to the challenges of Big Data and modern data science*’’ but also goes on to comment that ‘‘*the 6 high-level phases of CRISP-DM are still a good description for the analytics process*’’. These comments and the survey results may indicate that while a structured approach is valuable at a high level some level of customisation or flexibility is also desirable which is possibly why ‘‘My Own’’ is second in popularity.

For this purposes of this experiment, the CRISP-DM methodology was selected, as it gives structure to the overall data mining process, without being too prescriptive. The author is also cognisant of the need for flexibility as highlighted above and will adjust the methodology as required to achieve the desired outcomes.

3.2 Method

The iterative nature of a data mining process can be seen in Figure 1 below which shows the CRISP-DM data mining process. The arrows show the dependencies between each process/stage and the arrow going around the outside shows that as more knowledge is discovered the process may start over again whereby showing the ‘‘cyclical nature of data mining’’ (Shearer, 2000, p. 14)

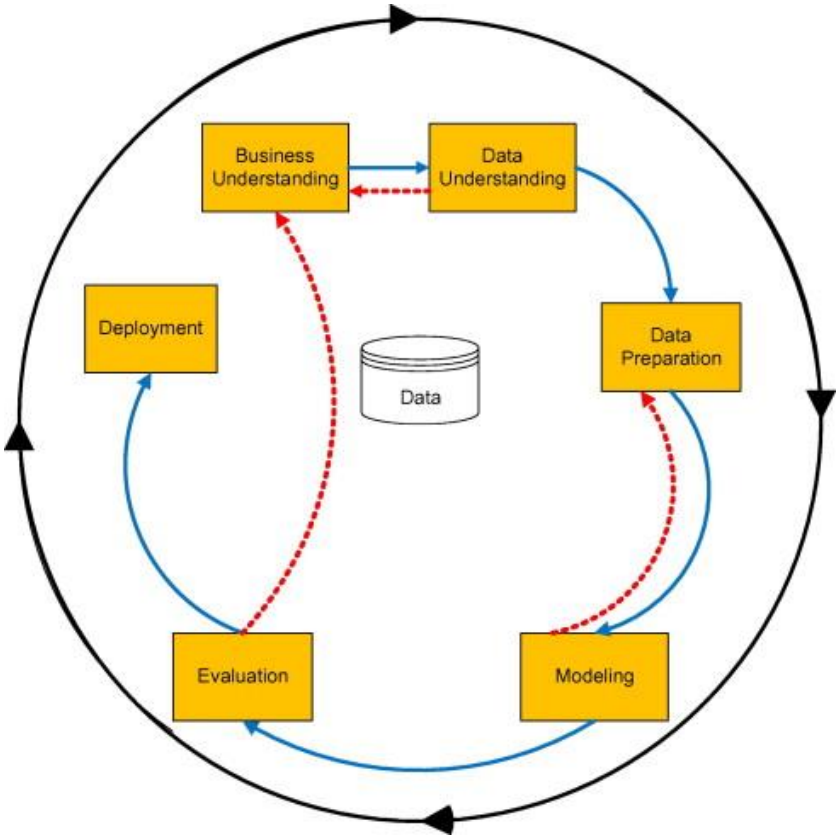


Figure 3-1 CRISP-DM process model (Sharma et al., 2012)

By mapping the dependant CRISP-DM steps into broader sections that interact with each other such as Business and Data Understanding, and Data Preparation and Modelling as shown in Figure 3-1, a broader more flexible methodology can be formed by which the experiment can be carried out and documented.

The approach taken in this data mining experiment will now be expanded on.

3.2.1 Business and Data Understanding

This section will cover both the ‘‘understanding’’ stages of the data mining process, business understanding and data understanding. Each of the ‘‘understanding’’ stages have a level of assumption to them, and as understanding is gained, these assumptions can be turned into knowledge that can be fed back into the data mining process.

‘‘Business Understanding’’ is largely driven by the main objectives and scope of this experiment, the data mining of log file streams for the detection of anomalies. This section will describe the outcomes desired from this iteration of the data mining process at a high level. Research question four places constraints on the goals of this experiment by making sure that it has the potential to form the basis of a practical and scalable solution. Data understanding will be built on over each iteration of the data mining process as knowledge is gained.

3.2.2 Data Preparation and Modelling

In this section log file attributes and the model or set of models that will give the best performance for the detection of anomalies will be proposed for evaluation in relation to the data mining experiment to be carried out.

Data sets will be selected and checked, based on the understanding/knowledge at the time. Each iteration of the data mining process will most likely have a smaller set of goals as outlined in the Business and Data Understanding section in order to build towards the overall desired outcomes of the experiment.

There are already some areas of prior art and expert domain knowledge that could be informative in this stage. It is envisioned that several models may need to be built and evaluated in order to find one that is suitable to meet the goals of this experiment. Modelling of the data will then be performed and the results recorded so they can be evaluated.

3.2.3 Evaluation

In this section the experimental results based on the data mining and modelling process will be presented. The results will be evaluated, contrasted and compared. As

a final part of the evaluation, a summary of the findings in direct relation to the research questions will be undertaken.

3.2.4 Summary

In the summary section, all four of the research questions will be discussed based on the results from the completed evaluation of the current data mining iteration. The knowledge gained from the evaluation as shown in the CRISP-DM data mining process can be fed back into the business and data understanding stages of the next iteration of the process thus completing the cyclic nature of the data mining process.

3.3 Chapter Summary

This section presented the methodology that will be used and how this was adapted and structured for use in this data mining experiment. Using this methodology and high level structure described by this method, the results of the data mining experiment will be presented in the next chapter.

There has to be a stopping point to the experiment, and this is bounded by the time allowed for the submission of this thesis. The CRISP-DM iterative data mining process will be completed until either a workable solution becomes apparent or time runs out. At this point the understanding and knowledge gained will be presented and future work highlighted.

Chapter 4 - Experimental Study

During the course of this data mining experiment four full iterations of the CRISP-DM process were achieved, these four iterations are presented in this chapter. As per the CRISP-DM process each iteration builds on the knowledge and understanding of the previous iteration.

4.1 Iteration One

4.1.1 Business and Data Understanding

The business objectives of this stage of the experiment had to be taken in context of the wider goals of this experiment, so at this early stage it was determined that it was necessary to take a closer initial look at the data and use domain knowledge to decide how it could be processed. Completion of this goal would lead to a deeper understanding of the data first hand. To achieve this initial insight and understanding of the data, an unsupervised approach was taken to explore the data further. By taking this approach the features of the data set were revealed and this led to further business and data understanding.

As this was the first stage and attempt of the data mining process, only a limited level of knowledge in direct relation to the objectives of this data mining experiment was already known. This knowledge about log files features originated from prior research and consultation with domain experts and it was clear that there was still many unknowns at this early stage of the experiment.

These unknowns covered a variety of areas from implementation of the tools required in order to collect and prepare the data, how to collate and store the data, and finally how to process and model the data. By researching and implementing these aspects of the experiment it allowed for the initial understanding and knowledge to be gained around how to achieve the overall experimental outcomes.

Through the implementation of the whole data mining process, business understanding could be gained in order to meet the wider goal of making a practical and scalable solution. The initial thoughts and decisions around tooling, data preparation and modelling methods are covered in the next section.

4.1.2 Data Preparation and Modelling

After evaluation of the data mining tools available and given the large number of unknowns around the data mining requirements, it was decided that a much more flexible approach to the data mining problem would be pragmatic. Thus rather than using a dedicated data mining tool which might limit the scope or approach taken, a custom coded data mining framework was created. By utilising a custom code-based approach, it supported the widest number of data mining processes possible for all stages of the CRISP-DM data mining cycle. This code-based methodology enabled a much more flexible approach to answering the research questions at hand and allowed for the end goal of finding a practical and scalable solution to data mining log files for anomalies.

After examining applicable languages and libraries, Perl and Python were both short listed as potential languages for the implementation of this work. After examining each language, Perl was selected. This was based on my own experience with the language and the large library of functions available to Perl which were directly related to the data mining approach of this experiment.

The first challenge was to receive the log information. By leveraging a standard Perl library, a simple log file collector was created using Net::Syslog (Howard, n.d.). This listened on the standard syslog port 514 and passed each log line into a function to process the data. This function then split the data into two parts, part one being the features based on the syslog standard, these syslog features are shown in Table 4-1 with the source listed as ‘‘SYSLOG’’, and part two being the actual message or log event.

In order to extract a further feature set from a log file message for further processing, a common method used by a number of researchers is the simple use of white spaced words and in some instances the position of the word is also utilised. This approach can be seen in Simple Log file Clustering tool (SLCT) by R. Vaarandi, 2003 and also utilised in his further work ‘‘Loghound’’ (Risto Vaarandi, 2004). This approach can also be seen in ‘‘A Lightweight Algorithm for Message Type Extraction in System Application Logs’’ by Makanju et al., 2012 in which they developed IPLoM (Iterative Partitioning Log Mining) which also uses white space to split each token (feature). Makanju et al.,

2012 then grouped each log line by message token count before processing them further based on the assumption that messages of the same token length are from the same print statement source and thus can be clustered on. This same type of feature set extraction is also seen in HMAT by Frei & Rennhard, 2008, where they not only used the word(token) count, but the character count/line length as well. Frei & Rennhard, 2008 also expressed interest in trying different approaches to feature selection as future work.

To further extend the predictor attribute feature selection of log files for testing, but still retain a generalised approach the Baler high level initial token generation was used (Taerat et al., 2011). This is shown in Table 4-1 with the source listed as "BALER". Baler did not just use space-delimited token generation as they stated that "it tends to give many and long token variations". This may indicate that by over-simplifying the feature selection too much it may limit the overall effectiveness of the solution by becoming over generalised in its approach and thus limiting its effectiveness to log files which only have white space delimited words in them.

In further examination of the approach that Taerat et al., 2011 took with Baler, it was decided to only use the initial feature set that is generated and not further tokenise based on English words as Baler does to build English word clusters. If like Baler, each English word was used it would also become another predictor attribute which would have to be stored and tracked. By taking the storage of each English word usage out of the scope of predictor attributes it allows for a much more generic solution as is desired but still leverages the log file features as introduced in Taerat et al., 2011.

HMAT (Frei & Rennhard, 2008) defined a word as being "any character sequence that does not include any white space." This aligns well with the work referenced previously but as described above may limit the effectiveness of this work. In contrast, in this work the sum of the total tokens per log line was used instead of the word count as defined in HMAT, this in principle should generate a higher count per line, but still should offer a general enough approach to be applicable across many log file line types without being overly sensitive to small changes. By using the token count it may even allow for increased detection as it may be able to detect feature changes that could be missed by the simple word count broken by white space approach. For the

character count the entire length of the syslog line was used. These features are listed with the source of ‘‘HMAT’’ in Table 4-1.

To extract this second set of features another Perl library HOP::Lexer - "Higher Order Perl" Lexer (Poe, n.d.) was used. This utilises regular expressions to tokenize the matched character strings into tokens as identified in Baler, generating a sum of each token type for each log line. These features are shown in Table 4-1 as Token Types.

Table 4-1

Token Types

Type	**Source**	**Description**	**Regular Expression or field type**
ALNUM	BALER	Alpha Numeric	`[0-9A-Za-z]`
ALPHA	BALER	Alpha	`[A-Za-z]`
DIGIT	BALER	Number (digit)	`[0-9]`
COLON	BALER	Colon	`[:]`
PAREN	BALER	Parenthesise(Brackets)	`[\[\]\{\}\(\)]`
SPACE	BALER	Space, Tab, Newline, Return	`[ \t\n\r]`
OTHER	BALER	Anything else not already matched	`[^:]`
LC	HMAT	Line Length (Count)	`Number`
TC	HMAT	Token Count (Word Count)	`Number`
LogTime	SYSLOG	Time sent	`Date/Time`
Time	SYSLOG	Time log received	`Date/Time`
IP	SYSLOG	Syslog IP address of sending server	`IPV4 Address`
Fac	SYSLOG	Syslog facility	`Number`
Sev	SYSLOG	Syslog Severity	`Number`

Table 4-1 Token Types

Following the collection of tokens per line, the results had to be stored somewhere. At this stage the most appropriate way to collect the results was unknown, so in order to make collection easy they were stored in a local database so they could be accessed

and examined as required. Two databases were briefly trialled in order to see which one would be the easiest to work with.

The two databases compared are easy to access and work with from Perl but take two very different approaches. The first database application that was examined was SQLite ("SQLite Home Page," n.d.). SQLite is a self-contained, server-less database which simply uses a file on a disk and thus is very easy to implement. This is in comparison to MYSQL ("MySQL :: The world's most popular open source database," n.d.). MYSQL is an open source relational DB product with commercial backing from Oracle. It comes in several variants, some which are free to use all the way up to massively scaled clustered instances. With this knowledge it was decided that although SQLite would be much easier to implement, MYSQL was picked in case it was necessary to scale the data storage application in order to achieve the desired outcomes of this experiment.

The initial data source was a university's central syslog collection service, this service was being setup at the time and permission was sought and granted to allow its use for the purpose of this experiment. The data source receives streams of log files arriving from a number of different systems and services, including but not limited to firewall logs, Microsoft AD server security logs, network security sensors, antivirus events and Cisco network logs. This effectively generated a stream of unsupervised data from a large real-world IT environment from a variety of log file sources for use by the data preparation and modelling tools.

Several design and build issues were discovered during the construction of the initial log processing/modelling tool, however with redesign and testing these were overcome. The two issues that stood out were the incorrect parsing of the log line information into tokens. The second issue was the overall performance of this initial approach to data collection.

Log parsing was confirmed with manual inspection of a sample of log lines to ensure that it was being performed correctly. A log parsing issue was discovered very early on, and was resolved in order to ensure correct generation of the predictor attributes for evaluation. It was found that the order of tokenising the information was critical to

ensure that the log line was processed correctly, once this was achieved the parsing was manually checked again in order to ensure correct operation.

An example of the parsing check can be seen in Table 4-2 below, where two sample log lines are compared and the token counts generated for each. Both samples in this case come from the Sendmail program, which is a common email gateway that is used across the internet.

Sample 1 "`Jun 16 10:23:53 mail1 sendmail[7701]: s5FMNo62007686: to=dl@mail2, delay=00:00:03, xdelay=00:00:00, mailer=smtp, pri=184004, relay=mail2.abc.ac.nz. [156.62.1.14], dsn=2.0.0, stat=Sent (s5FMNrsR019233 Message accepted for delivery)`"

Sample 2 "`Jun 16 15:08:57 mail1 sendmail[30650]: s5G38Hwn030124: s5G38r27030650: DSN: Remote protocol error`"

Table 4-2

Token Log Sample Token Counts

Type	**Sample 1**	**Sample 2**
ALNUM	5	3
ALPHA	20	6
DIGIT	19	5
COLON	8	6
PAREN	6	2
SPACE	19	10
OTHER	25	0
LC	230	97
TC	102	32

Table 4-2 Token Log Sample Token Counts

The overall performance of the log processing software presented another serious issue related to the way it was implemented. The custom code-based Perl log processing software ran on the log collection server and wrote results to a remote database. The volume of streaming log file data, while not huge, was still large enough

to cause performance issues for both the log file processor and remote DB. As an immediate workaround it was decided to change to a batch insert type model and insert groups of 100 log lines at a time to work around this problem. This approach largely solved the issue at this stage of the experiment, and allowed for the collection of an initial set of results to be achieved.

In summary, for this data preparation and modelling section of this iteration, the attributes to be used were defined and are presented in Table 4-1 earlier and the source of the log files/data was determined. The predictor attributes were extracted and stored in a database for examination and evaluation.

4.1.3 Evaluation

The experimental data will now be examined in order to increase understanding and knowledge so any findings or discoveries can be fed back into the next iteration of the CRISP-DM data mining process.

Figure 4-1 shows an example of a typical run. This sample was taken over a 45 min period in which 374517 log lines were recorded, equating to approximately 8323 log events per minute or 138 events per second (EPS). This represents a typical steady state of the log collector around 3:00 pm at the time this data was taken.

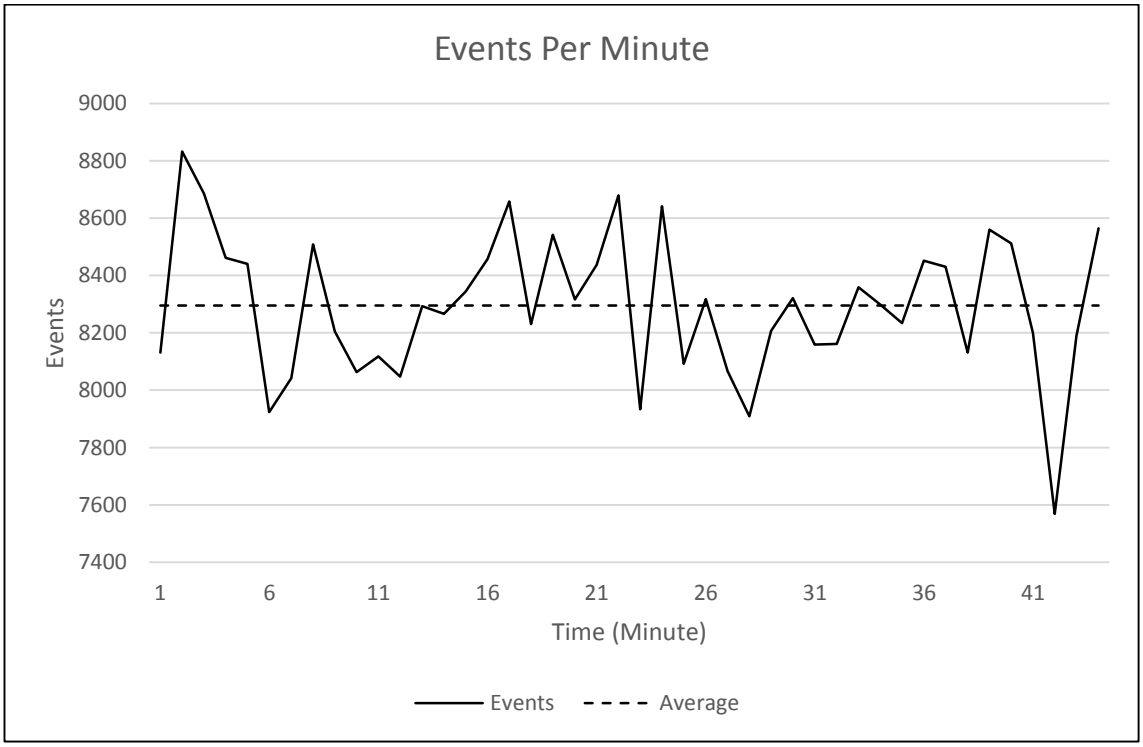


Figure 4-1 Events per Minute

Given the performance impact that it was placing on the system in its current implementation, only short runs were performed at this stage to limit any negative effects on the production log collection system.

In the article presented by Butler, 2009 ‘‘Benchmarking Security Information Event Management (SIEM)’’, he described a medium sized organisation of 750 to 1000 users along with the typical log events they generate across a number of systems and services. He calculated a steady state approximately 150 EPS, given that not all the universities systems or servers were being logged in the production system under testing at the time the experiment was done this baseline rate could be considered comparable with the current findings.

While this steady state rate is similar, Butler, 2009 notes other EPS rates that should also be taken into account. He presented his estimations for a medium sized organisation, these are a potential peak of 15600 EPS with an average Peak of 8100 EPS. As can been seen in Figure 4-1, while there was some variation around the average, there were no large spikes to indicate an event taking place which would push the EPS rates into these much larger ranges.

Other solutions have been built which can collect log data at these high rates, such as Chukwa (Rabkin & Katz, 2010) and Logjam (Cardo, 2011). Thus there are alternatives which could be investigated further if data collection at high speed is an ongoing problem. It should also be noted that Chukwa also offered methods for processing the data for feeding into an anomaly detection system as well.

Given the performance issues noted during the design and build phases, special attention will have to be taken to ensure that the data preparation and modelling framework is designed in such a way that these large peak EPS rates could be processed without adversely affecting any other system or the desired outcomes of this experiment.

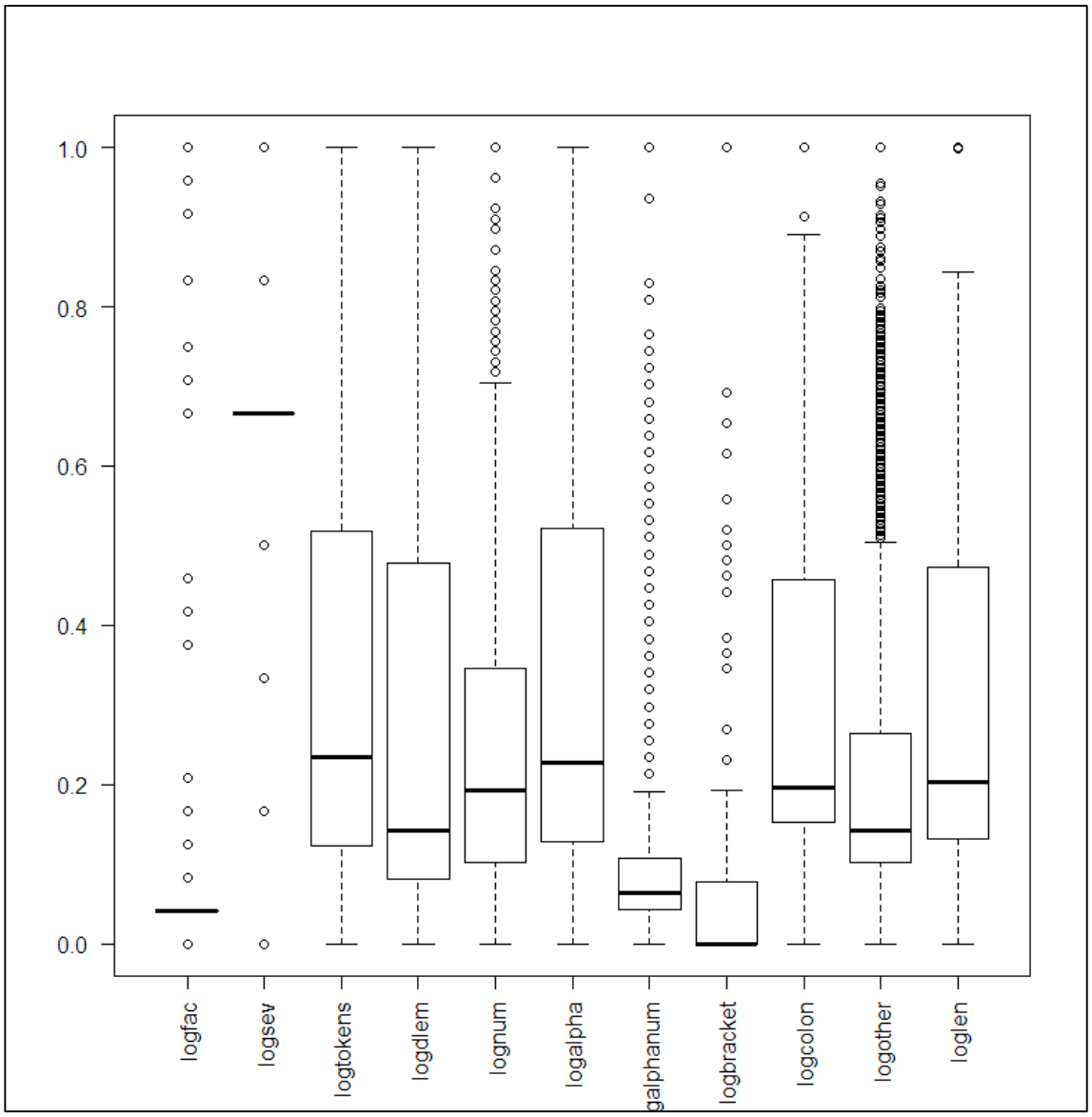


Figure 4-2 Predictor attribute box plot

Figure 4-2 shows box plots of each predictor attribute that was collected. They were generated using the R environment (R Core Team, 2015) with each range normalised from 0 to 1 so they could be compared directly against each other. The box plots used the default Tukey outlier definition of 1.5 x inter quartile range (IQR) with the outliers shown as circles.

Of immediate note was the two syslog predictor attributes, logfac which is the syslog facility and logsev which is the syslog severity level. As can be seen in Figure 3 they have almost no variance and thus generate a large number of outliers. Given this lack of variance, if a similar anomaly detection measurement technique to the one used in HMAT is employed it could generate a large number of false positives which could mask variances from the other predictor attributes. As noted in the introduction, the anomaly detection that this experiment is trying to achieve is not the one off event which is much more easily detected by signature or rule based approaches, but more when the "mix" of predictor attributes has changed.

This can be contrasted with the other attributes which can be grouped by the range of outliers they exhibit. With no outliers in this sample set, logtokens (Tokens), logdlem (Delimiters), logother (Other, anything not matched) and logalpha (Alpha) have good variance. Still displaying good variance but with some outliers lognum (Numbers), logcolon (Colons) and loglen (Line Length). This leaves logalphanum (Alpha Numeric) and logbracket (Brackets) which only show a small variance and a large number of outliers.

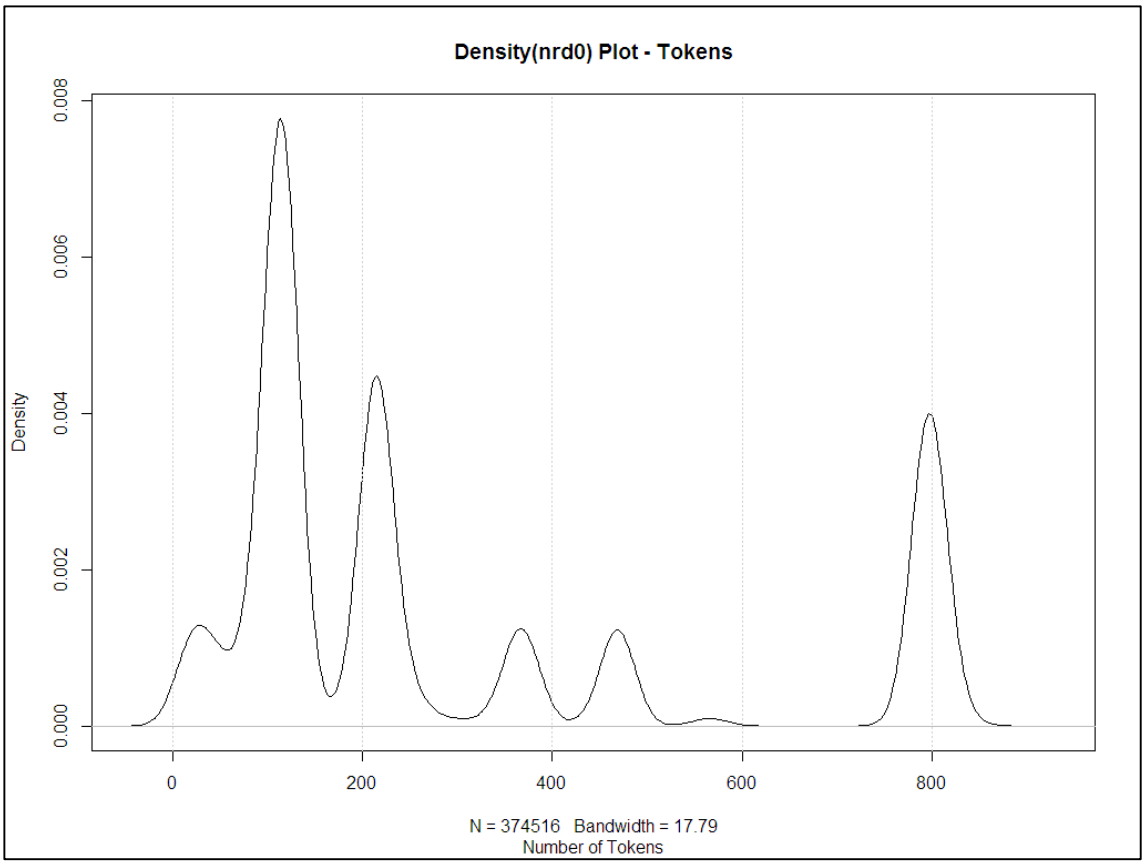


Figure 4-3 Tokens Density Plot

Figure 4-3 shows an ‘‘R’’ density plot (“Kernel Density Estimation,” 2015) using the default Gaussian kernel with the smoothing bandwidth selected by the rule-of-thumb ‘‘nrd0’’ method of the token count across all instances of the 45 minute sample. It can be seen that it is a non-Gaussian distribution and is multimodal.

The multimodal nature of the distribution could be derived from the nature of the log source data as it is the representation of a number of log sources consolidated into one. Even with logs source from one system, different message types could have very different token counts depending on how they are reported to the logging system.

Also of note is that from this sample period there were some peaks which are much larger than others, this may have an effect on how anomalies are detected and a very large peak may mask the detection of a change in one of the smaller ones.

4.1.4 Summary

A summary of the results of the evaluation will now be presented in context to the research questions at hand and this will provide feedback into the data mining process for the next iteration.

Question One – **Efficient and accurate predictor attributes**

While syslog was a convenient log message transport protocol, the predictor attributes specific to it (Facility and Severity) don’t appear to offer much value as they are very specific to the syslog log message format and are not widely used across all log message types as can be seen by the very low variance. In the end solution it is not the author’s intention to keep each and every line as a separate log entry, rather the author intends to generate a summary from which anomaly detection can be run against, so the line specific attributes (Log line Time and IP) should also be removed as they will not be able to be used in a generalised fashion and are more suited to other anomaly detection types which are based on log volume and source.

Frei & Rennhard, 2008 also supported this and noted that while volume is important it does not contribute to the anomaly detection. They were more concerned about whether the mix of log messages had changed and this is what drove their anomaly detection level.

The low variance shown among some of the other predictor attributes is most likely largely influenced by how the log file messages are generated and how the individual author of that program has decided what their output format should look like and how it should be recorded. As noted in the literature review there is still no one standard for all log format types, and while this is the case a more generic approach to the predictor attributes should be considered as an acceptable approach that could be applied broadly to many different log format types. Also focusing on just one predictor attribute might not be wise. In HMAT where they have picked white spaces, it limits the approach to log files which have a variance in white spaces only. Some log files may not have any variance in white spaces and thus having other predictor attributes may allow this limitation to be overcome.

After discussions were held with a domain expert, an initial time slice of one minute was discussed and presented as a sensible time interval between checks. As shown in Figure 4-1 this currently equates to an average of 8323 log events per minute. This offers a balance between sample size and potential speed of detection. Given that this is to be used in a streaming type environment, a one minute window is still an acceptable time frame for an anomaly detection system to function within, while still offering the ability to be altered if required and not use out-of-band log processing to find it after the fact.

Question Two – Performant model/s for the detection of anomalies

No anomaly detection models were evaluated in this first iteration, but the predictor attributes were. As can be seen from the box plot in Figure 4-3 some outliers can already be detected. If these are true outliers or just an outcome of the mix of different log file types all being combined together is still yet to be determined.

Also in order to get further understanding of the data as intended, every log event was recorded and this approach is more typical of a signature type anomaly detection methodology. While each line by itself is quite unique, the aggregate mix of events over a period of time is intended to be used.

Question Three – Evaluation of anomaly detection model/s.

From now the goal is to find a distance/change detection type model or models that could be employed and are suitable for the log file predictor attributes that have been selected. This moves towards a more generic application and approach as influenced by HMAT (Frei & Rennhard, 2008) and Baler (Taerat et al., 2011) now that data knowledge and understanding has been increased.

In order to narrow the focus initially to allow for discovery of model types in the next iteration, anomaly detection will be focused on line length, this predictor attribute showed good variance, and did have some outliers and thus offers a middle ground between the attributes with no outliers and ones with many.

Question Four – Practical and scalable performance constraints.

As noted in section 4.1.2 several performance issues were apparent in the initial design in relation to the volume of data that was being processed. An initial work around was performed by changing to a batch insert design in order to achieve the initial goals of this stage of the data mining experiment but further work was identified as necessary to overcome these issues.

The design change required for the next stage of this experiment should overcome many of these initial issues as it moves to a more generalised approach and is not trying to process everything one line at a time into a remote data store.

With the design change to a one minute window in the next iteration, an update to a database will only need to occur once every minute and this should resolve any database transaction type issues. While this takes care of the transaction speed side, ensuring the client application is performant is also of concern. With this in mind the single threaded initial application will be modified into a multithreaded one to ensure that data can still be collected while it is writing to the database.

At this time it was also considered that if this multithreaded approach could not be achieved, then it might be possible to leverage another log collection system to perform this aspect of the framework allowing the anomaly detection to be handled independently of it. It is not intended to pursue this at this time, but will remain in mind as part of the wider outcomes of this project.

Incorporating some early form of visualisation at this stage may also offer further insights into the data, thus increasing knowledge and understanding. At this stage the unsupervised data approach is still appropriate to achieving the goals of this experiment and thus will be continued with into Iteration Two.

4.2 Iteration Two

4.2.1 Business and Data Understanding

The business objectives of this stage are to build on the knowledge and understanding gained in the first iteration by starting to investigate distance models and visualisation techniques in order to progress the overall objective of the experiment. The distance models are to increase the business and data understanding in relation to the detection of anomalies. In addition to the distance models, visualisation methods are also to be explored. By achieving both of these goals it will offer key insights into potential anomaly detection methodologies.

There were a number of issues that were discovered during the first iteration of the data mining experiment. In order to tackle these issues, during this second data mining iteration a number of modifications needed to be performed to the design of the framework. The main changes to the framework was the move to a one minute window for each sample of the data and increasing the performance of the logging client by moving to a multithreaded model.

Visualisation of the data was explored in order to increase data understanding and offer other insights into the data. In order to allow for this to be used going forward web based technologies were utilised, so if necessary it can be developed further into a practical solution.

4.2.2 Data Preparation and Modelling

The syslog specific predictor attributes ‘facility’ and ‘severity’, while still being recorded, will not be evaluated any further in this and future iterations. Additionally with the move to a one minute sample interval any log line specific attributes were removed or summarised. The log line specific attribute that was removed was the IP address of the originating server. The time that the log line was received was summarised and recorded when the log line summary of predictor attributes was sent

(triggered every minute with the servers all being synchronised by NTP). The attributes that were recorded for each entry are Year, Month, Date, Day of the Year (1 <-> 366), Weekday (0-Sunday <-> 6-Saturday), Hour and Minute. Evaluation of attributes during this iteration was focused on log line length.

A practical approach to histogram bin sizes was used. Without knowing the level of detail that a distance model would require, a range of bin sizes was used with a lower and upper bound based on the approximate range of values. If necessary other bin sizes could be arrogated from the ones selected by running a function across the data to make a bin size a factor of one of the bin sizes selected.

Bin sizes of 1, 5, 10 and 20 were selected and applied based on the length of the data observed. Attributes with a small range of values had small bin sizes applied (1, or 1 and 5) and attributes with large ranges had the large range of bin sizes applied (5, 10) or (5, 10 and 20) thus trying to balance the size of the number of histogram bins against any loss of precision.

After initial tests using a database approach to storing data caused performance issues other approaches were also evaluated. Generating a database schema to store a histogram type data set efficiently and also having the ability to perform data aggregation functions on it was found to be challenging. In order to offer a flexible and scalable framework going forward other information storage options were investigated.

One approach that could solve these issues was moving from a relational database type approach to a document store which used key/value pairs. This type of approach was used successfully by Liu, Pan, Cao, & Qiao, 2010 in their paper "System Anomaly Detection in Distributed Systems through MapReduce-Based Log Analysis." This type of MapReduce data store is commonly implemented in NoSQL type systems. Several NoSQL variants were evaluated based on ease of use, scalability and support for key/value pairs for MapReduce operations and in the end CouchDB by the Apache Foundation was selected. CouchDB supports a web orientated API which also aligned with the current visualisation approach being taken as well. By introducing MapReduce functionality it also allowed for further data aggregation operation to be performed as required. In future applications data aggregation could be achieved by aggregating

each time recorded into larger time slices, for example for each one minute sample it could be aggregated into 5/10/15/30/60 minute time slices as required by adding the appropriate histograms together using a MapReduce function etc.

In addition to this a multithreaded client was developed to handle the volume of the log files being generated and updated to the NoSQL document store. By moving to a multithreaded model, update time was no longer an issue as data collection could continue while the information for the last minute was being sent to the appropriate NoSQL server.

With the data collection issues now resolved, distance/change detection type models were now required to be tested and evaluated in order to achieve the major outcome of this work, anomaly detection in log file streams. The tests were based on the log line length initially from the predictor attributes in order to give some focus to finding a practical anomaly detection model. To start this testing the previous summary histogram is used as a baseline for comparison with the current summary histogram.

After consultation, model evaluation was initial started with the Kolmogorov–Smirnov (KS) test in order to measure the differences/distance between both of the samples (previous to current) and over the course of this iteration Kuiper's test was also added for evaluation to change the sensitivity location of the test ("Kolmogorov-Smirnov and Kuiper's Tests of Time Variability," 2012).

4.2.3 Evaluation

A simple visualization solution was created which displayed a visual representation of the log line length histogram, this was generated by functions written in the "Processing Visualization Language for the web", ProcessingJS ("Processing.js," n.d.). The effective probability calculated from a sample of the evaluated results was also compared manually to double check results.

A shade of green was used to show bin size, bright green for the largest bin (the largest bin was normalized to the brightest green) gradating down to black for the smallest. In addition, on each bar a line was drawn for the calculated probability that they came from the same distribution for both the Kolmogorov–Smirnov (Red) and Kuiper (Blue). The top of each bar represents 100% and the bottom of the bar 0%.

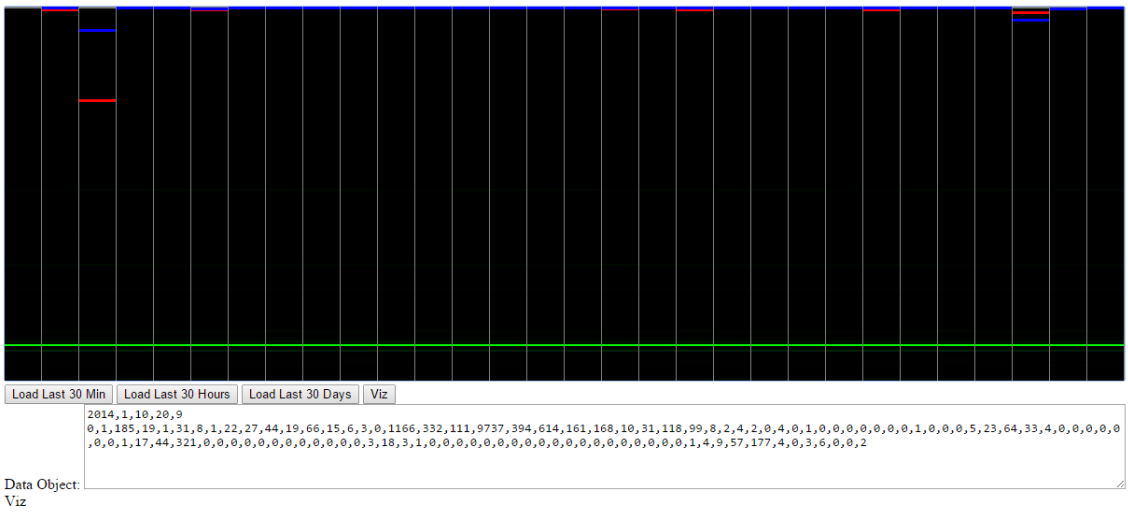
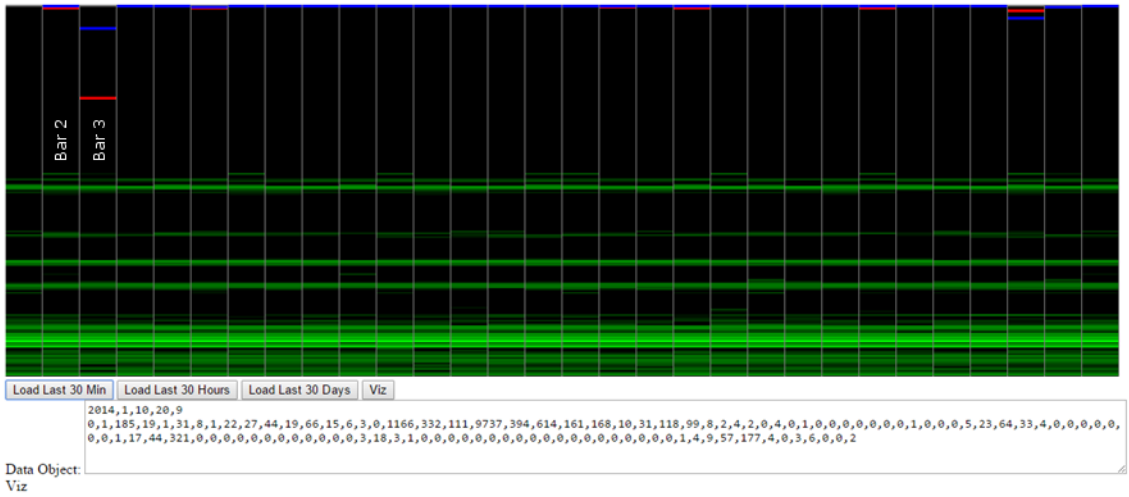


Figure 4-4 Sample Visualization

When first implemented, that range between the largest bins and the smallest was so large that all that was clearly visible on the bar was the largest bin as shown in Figure 4-4. The largest bin in this case was 9739, this can be compared to the next largest bin size 1166 which can be seen as a faint line underneath the bright green one. In order to give more visibility to the data and cope with the large range of bin sizes the log of each bin was taken and then this was then used to display the bin values using the shades of green as intended. This new result can be seen in Figure 4-5.

Initial results from the Kolmogorov–Smirnov model showed that this model was overly sensitive to log lines that had a large change in the mean due to a change in the log line length, but when manually compared the overall mix of messages was still quite similar. Given the generalised approach that is desired, this over sensitivity to one particular predictor attribute's statistical change rather than the mix might have too strong an influence on the overall results.

To overcome this sensitivity to changes in the mean, the Kuiper test was introduced, it is still directly related to the Kolmogorov–Smirnov test but is not as sensitive to changes in the mean, and spreads the sensitivity to the tails ("Kolmogorov-Smirnov and Kuiper's Tests of Time Variability," 2012).



*Bar 2 and 3 have been labelled for ease of reference

Figure 4-5 Sample Visualization (Log)

This is the same data as in Figure 4-4 but now with the log of each bin being taken before it is processed for visualisation. As can now be seen by the increase in the shading across the range of bars, there is now much better visibility to the data and it enables the smaller bin sizes to be shown on each bar thus increasing the level of knowledge and information that the visualisation is presenting to the user.

This result could also indicate that by taking the log of the bin values, it may allow for greater sensitivity of the smaller deviations in a predictor attribute to have more influence on the overall result depending on the anomaly or change detection model used.

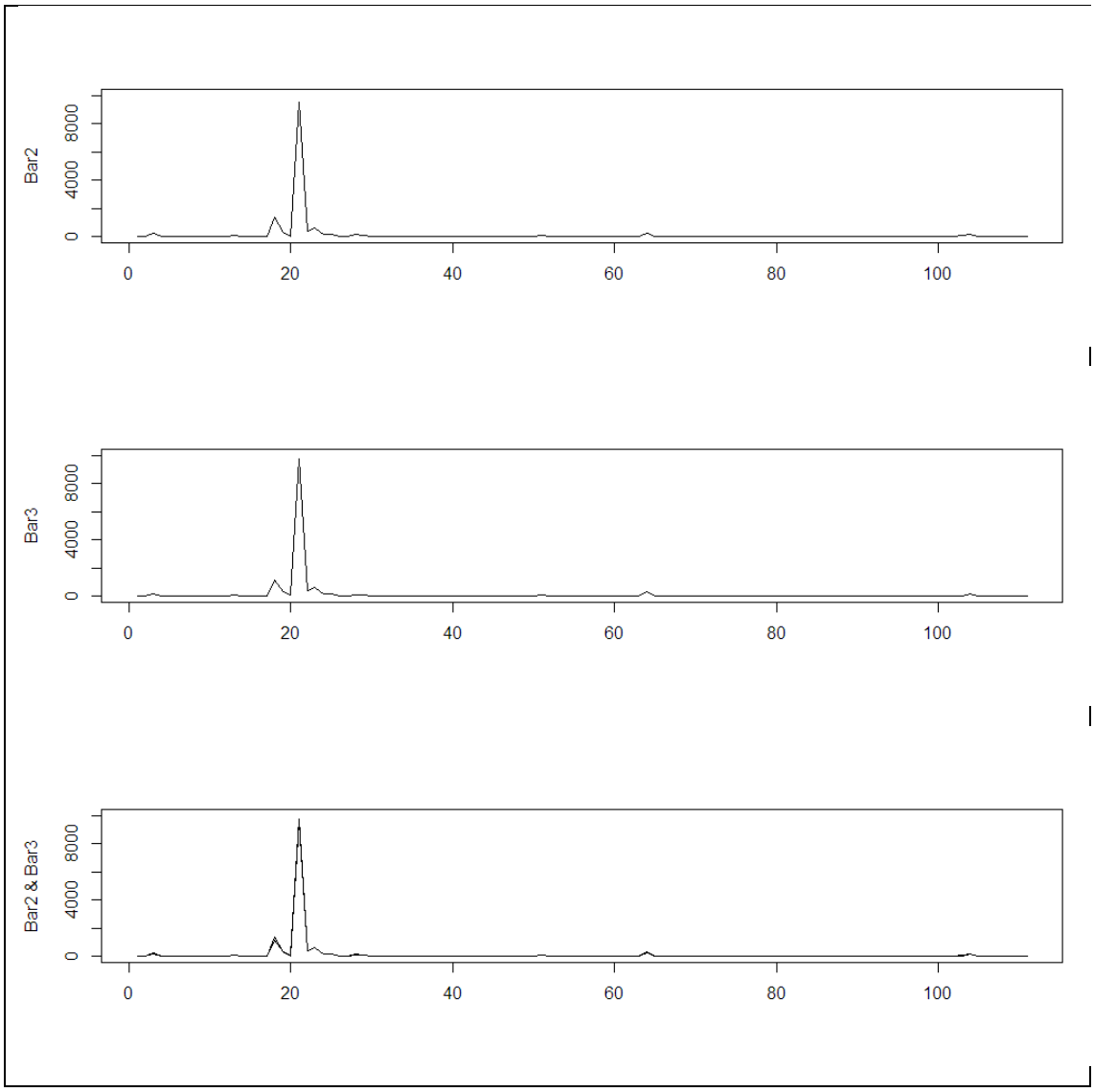


Figure 4-6 Visual Comparison

Evaluation of the detection models was performed by a visual review of the test results. In Figure 4-5 the data that generated ‘‘Bar 3’’ was compared to the data that generated ‘‘Bar 2’’ as both the Kolmogorov–Smirnov (KS) and Kuiper's tests showed a relatively large difference when compared to the other results. The data sets were then plotted as shown in Figure 4-6. A visual examination of Figure 4-6 Bar 2 and Bar 3 data shows little difference and the overlay plot of Bar 2 & 3 data shows directly that there is very little difference visually between them at all . This appears to be inconsistent to the results generated by the Kolmogorov–Smirnov (KS) and Kuiper's tests where a p-value of 0.76 was calculated. This p-value indicates that it is highly

unlikely that the distributions are similar. This may indicate that the Kolmogorov–Smirnov (KS) and Kuiper's tests may not be suitable for this experiment and that further testing and evaluation needs to be carried out.

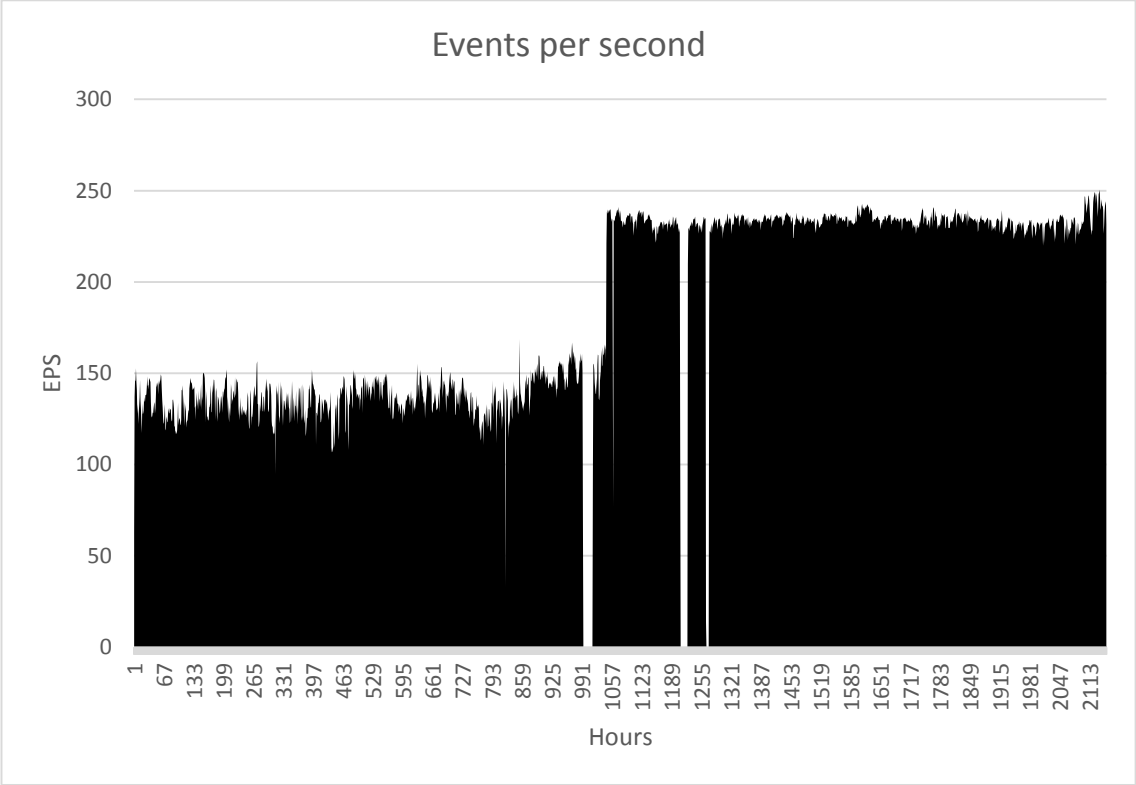


Figure 4-7 Events per second

The EPS as shown in Figure 4-7 shows that the performance issues encountered during the first iteration have been resolved due to the implementation of a multithreaded approach to the log processer and by the change to CouchDB as a data store. Figure 4-7 shows approximately ninety days of log predictor attribute information aggregated each hour from one minute samples as collected from a university's central syslog server. No performance issues were encountered over the 90 day period.

The gaps in the graph are times when the central log service was stopped during upgrades and maintenance. Also additional log sources were added approximately halfway through the collection period which increased the rate by about 100 EPS, displayed as a step in Figure 4-7. In a closer examination, when looking at each minute of the data collected a maximum rate of 466 EPS was recorded. While this is much lower than the rates predicted by Butler, 2009 it still shows an almost two fold

increase from the average EPS rate over the last half of the sample set. It is noted that during this period no extreme events occurred which could have caused any spikes in volume, but this potential to spike to high volumes for a log source should still be taken into account.

4.2.4 Summary

This data mining iteration focused on solving the performance issues found during the first iteration, and started the evaluation of anomaly detection models using the log line length predictor attribute. It also introduced a simple visualisation method using web based technologies.

Question One – Efficient and accurate predictor attributes

During this iteration, focus was on log line length as a predictor attribute. No change to the current log file predictor attributes were identified during this iteration of the experiment. While the attributes themselves appeared to be working well, complexities around how often to sample and how to bin data efficiently and accurately for the histograms were highlighted and will need careful consideration as to how they will need to be managed in the overall framework.

Moving to a one minute window removed or summarised all the individual log line specific attributes, this was an expected outcome and moves the framework to a more generalised approach for anomaly detection in streaming log files. Each group of predictor attributes recorded had summary time stamp information added which allows for MapReduce type operations which could be used to further summarise the information across different time slices if required.

The current one minute window may have limitations around slow data rates, it should be considered that there may be no one correct way to time slice the data for generation of predictor attribute histograms for comparison, but rather a flexible range of options should be made available. This could include options such as a minimum number of samples to collect before anomaly detection is performed. Once again, the use of MapReduce type operations across the time may facilitate this type of approach.

Multithreading and summarising log line predictor attributes into one minute windows worked well for the data rates that were active during this testing phase. A one minute window still allows for a timely response if required from a streaming data source as well as not creating a performance demand on the overall system itself.

Taking the log of each bin value as shown in Figure 4-5 allowed the smaller features of the histogram to become visible, this could also affect the performance of any anomaly detection type models used and should be evaluated further as part of the data preparation.

Normalisation of the data using the peak bin value of each histogram to compare allows for the direct comparison of histograms as the only difference expected is the mix of log line types, not the volume of data. This is the approach taken in HMAT, and this approach will be continued further as part of the data preparation.

Question Two – Performant model/s for the detection of anomalies

There are many ways to establish a baseline for comparison, sliding window as used by Mohammadjafar Esmaeili & Arwa Almadan, 2011 or even a seasonal model such as Holts Winters (Winters, 1960). For a large batch process there will probably be an ongoing sequence of events that happen continually over a period of time for a set time interval, with no seasonal effect to this, it just starts and finishes. This could be compared to a system that has a large user population that has a daily start and finish time and heavy seasonal variation.

A method that could support both of these activity types is probably the simplest of the window types which just compares the current sample to the previous. While this would miss a gradual change over time and seasonal effects, something that the Holts Winters or sliding windows might be better at detecting, it would still have the ability in both cases to spot a short-lived or stepped change as an anomaly event that happens between samples as both event types will cause a change in the log file attribute mix from normal activity.

During this iteration two distinct goodness-of-fit/distance type models were compared, the Kolmogorov–Smirnov (KS) and Kuiper's tests. Both of these tests are related as they are based on cumulative distribution functions to test the uniformity of

a set of data distributions. Their point of difference is that they are each sensitive to different types of changes in the distribution function, KS is sensitive to changes at the mean, and Kuiperʾs is more spread to include sensitivity to changes in the tails. The inconsistent results discovered when examining the results as shown in Figure 4-7 implies that other distance and anomaly detection models will also need to be tested.

Question Three – Evaluation of anomaly detection model/s.

The result of the Kolmogorov–Smirnov (KS) and Kuiper's tests are a probability value that the data from two windows came from the same distribution. This is a useful measure of the difference, however it is not that easy to compare the amount (size) of difference across a range of results. It is expected that, like in HMAT, there will be some range in variation of the mix of message types which could be considered normal, and methods to quantify this variation will be experimented with in a further iteration.

The models tested so far do not achieve the outcomes that this project was looking for. More models will need to be explored and tested in the next iteration in order to find a model that will more directly meet the goals of this experiment. Thus the problem of achieving a more quantifiable distance measurement, which allows for a direct interpretation of the measurement across a range of results will be addressed.

Question Four – Practical and scalable performance constraints.

The multithreaded approach to the log processer and the change to CouchDB as a data store solved the performance issues. This distributed NOSQL approach offers the ability to shard and aggregate (MapReduce) across a number of collection nodes and time slices as needed for comparison, so it is possible that a framework developed with these as a base could be deployed within big data type environments and be a performant solution.

During this iteration, a lot of time was spent creating a generalised visualisation for the overall solution. While this did offer some new insights into the data, such as the value of taking the log of the histogram bins in order to bring visibility to the large range of bin values, it did require a lot of time which may have been better spent with more in-depth evaluation of the data and models at hand.

4.3 Iteration Three

4.3.1 Business and Data Understanding

The business objectives of this stage are to build on the knowledge and understanding gained in the second iteration. The distance models used in the second iteration did not achieve the desired outcomes expected so the focus of this third data mining iteration was the evaluation of further distance/anomaly detection models using the current set of predictor attributes, once again with a focus on log file line length.

In this third iteration a major change to the data mining approach was required, this was the change from an unsupervised data set to a supervised one. This change in data set was used to focus this iteration on the performance of the anomaly detection models being evaluated. In order to facilitate this change in approach it was necessary to build an automated data generation and testing framework.

For this iteration the author consulted a domain expert with over twenty years of experience in dealing with UNIX systems administration and the extensive use and analysis of log files in order to diagnose system errors and issues. From his domain knowledge he identified possible log files to use for evaluation and a mix of good and bad log events for each (see Appendix A.)

As introduced previously, it can be considered that a system or service has a typical pattern of log events associated with its operation. For example on a DHCP (Dynamic Host Configuration Protocol) server where machines are left on for a long period of time there will be many pairs of ‘‘good’’ DHCPREQUEST and DHCPACK log messages, but in an error state the mix of messages will change, one such ‘‘bad’’ message could be a DHCPDISCOVER message with ‘‘no free leases’’ as a log entry repeated over and over again. This changes the mix of log events collected by a predictor attribute and is what should be detected as a change and thus an anomaly.

4.3.2 Data Preparation and Modelling

The change to a supervised data set required a change in testing methodology for evaluating distance models. The supervised data set was constructed from 10 good log lines and 10 bad log lines as shown in Appendix A for a representative sample for each system type. A data set was then generated based on combining various quantities of

‘‘good data’’ and ‘‘bad data’’ to form a combined complete data sample set of 1000 log lines. This data set was then compared to a baseline set of all good log line data. A range of data sets were generated by stepwise increasing the ratio of good and bad data to give a total of 100 different data sets ranging from all good to all bad. This effectively gave a 1% step change in log file mix for each increase in the ratio of good to bad.

Three different log file types were selected for use in evaluating the anomaly detection models being used.

The log types selected were:

- SendMail logs
- SSH logs
- DHCP logs

These new data sets and testing methodology allowed for a more direct comparison of the various distance models and also a way to compare their sensitivity to change. This greatly improved understanding of the data and how different predictor attributes are affected in relation to the mix of good and bad being altered.

In addition to moving to a supervised data set, data preparation methods were also tested. The methods tested were the use of the actual vs the log value of each bin as identified during iteration two. This allowed testing of the hypothesis formed in iteration two, this hypothesis being that by taking the log of the bins before input into the distance models, it will make them more sensitive to changes of the smaller bin values allowing anomalies or smaller feature changes in each data set to be better detected. In addition all datasets were normalised before having their distance calculated to remove any peeks and to align with the findings in this experiment and other research around volume normalisation.

A further review of literature was undertaken guided by the greater data and business data understanding now gained. It was discovered that a field of research that might be useful and also employs histogram distance measurement type techniques to detect matches (and thus differences as well) is the field of image pattern matching and detection. One example of work in this field is the ‘‘The Earth Mover's Distance as

a Metric for Image Retrieval" by Rubner, Tomasi, & Guibas, 2000. In this work they identified a set of new image features (histograms) and then utilised the earth mover's distance as a metric to find similar images.

Further research in the field of image processing by Rubner et al., 2000 contrasted and compared several other histogram distance measurements including Kolmogorov–Smirnov, Quadratic-form distance and others. The Earth Mover's Distance measurement performed well in their experiments when used in conjunction with the image features they extracted. Given the successful use in this and other image processing research papers the Earth Mover's Distance measurement was selected as one of the new distance models to try during this iteration. An additional method was selected for testing, the Quadratic-form distance. This has been widely used in the past as a histogram distance function but takes a different approach as the "Quadratic-form distance does not enforce a one-to-one correspondence between mass elements in the two histograms" (Rubner et al., 2000) and thus offers an comparative method to the Earth Mover's Distance measurement.

There are a number of ways to implement an earth mover distance function and two methods were selected as they both implement the bin distances and weighting by contrasting methods.

The following pre-processing was carried out on all data sets before any model distance evaluation was performed

For array A and B Normalise A and B to 1 Make A and B the same length by appending 0 to the shortest one

The models selected to be evaluated for this supervised iteration were:

Kolmogorov–Smirnov

Kolmogorov–Smirnov test and probability measure, code and test adapted from the "Root" TMath library from Cern "TMath:KolmogorovTest," n.d. accessed at *https://root.cern.ch/root/html/TMath.html#TMath:KolmogorovTest* under the LGPL

licence and description from “Kolmogorov-Smirnov and Kuiper’s Tests of Time Variability,” 2012

$S_N(x)$ is the cumulative distribution function of the probability distribution from which a dataset with N events is drawn. If N *ordered* events are located at data points x_i events... , N, then:

$$S_n(x_i) = \frac{i - N}{N}$$

Comparing two different cumulative distribution functions $S_{N1}(x)$ and $S_{N2}(x)$, the KS test statistic is:

$$D = \max_{-\infty<x<\infty} |S_{N1}(x) - S_{N2}(x)|$$

Probability for the Kolmogorov–Smirnov test is given by:

$$P(z) = 2\sum_{j=1}^{\infty}(-1)^{j-1}e^{-2j^2z^2}$$

The returned value probability is a calculated confidence level which gives a statistical test for compatibility of A and B. Values of probability close to zero are taken as indicating a small probability of compatibility. (“TMath:KolmogorovTest,” n.d.)

Kuiper

Kuiper tests and probability measure, code and test adapted from “aarchiba/kuiper,” 2009 accessed at *https://github.com/aarchiba/kuiper* under the scipy license. The original code for the Kuiper statistic and probably calculations was derived from formulas defined in Paltani, 2004.

Using the same cumulative distribution function as the Kolmogorov–Smirnov test.

The Kuiper test statistic is defined as:

$$V = D_+ + D_- = \max_{-\infty<x<\infty}[S_{N1}(x) - [\ S_{N2}(x)]] \ + \ \max_{-\infty<x<\infty}\left[S_{N2}(x) - \ [S_{N1}(x)]\right]$$

From this statistic, the function as described by Paltani, 2004 gives the probability of obtaining two samples this different from the same distribution is returned.

Quadratic Form Distance

Adapted from "stat_analysis.pl," n.d., accessed at *http://ftp.filesystems.org/osprof/analysis/stat_analysis.pl* under the GPL licence

The Quadratic-form distance is defined as:

$$\mathrm{D} = \sqrt{(\mathrm{h} - \mathrm{k})^T \times \mathrm{Z} \times (\mathrm{h} - \mathrm{k})}$$

Where h and k are vectors that list all the entries in A and B. Cross-bin information is incorporated via a similarity matrix Z where zij denote similarity between bins i and j . Here i and j are sequential (scalar) indices into the bins. For this work the recommendation in Hafner, Sawhney, Equitz, Flickner, & Niblack, 1995 was used where zij = 1 - (dij/dmax) where dij is the ground distance between bins i and j of the histogram, and dmax = max(dij) "stat_analysis.pl," n.d.

Earth Mover's Distance (EM) – Implementation 1 (Weighted by histogram total)

Adapted from "stat_analysis.pl," n.d., accessed at *http://ftp.filesystems.org/osprof/analysis/stat_analysis.pl* under the GPL licence

The Earth Mover's Distance (EM) is defined as:

$$EM_0 = 0$$

$$EM_{i+1} = EM_i + \left(\frac{A_i}{\sum A}\right) - \left(\frac{B_i}{\sum B}\right)$$

$$TotalDistance = \frac{\sum |EM_i|}{|A|}$$

(See Appendix B for library code function)

As A and B are the same length only the length of A is used to divide the result by.

Earth Mover's Distance (EMD) – Implementation 2 (One to One)

Adapted from "EarthMoversDistance (Apache Commons Math 3.6-SNAPSHOT API)," n.d. under the Apache License, Version 2.0 accessed at *https://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/ml/distance/EarthMoversDistance.html#EarthMoversDistance()*

The Earth Mover's Distance (EMD) is defined as:

$$EMD_0 = 0$$

$$EMD_{i+1} = (A_i + EMD_i) - B_i$$

$$TotalDistance = \sum |EMD_i|$$

(See Appendix B for library code function)

4.3.3 Evaluation

Using the supervised data set of log lines, results were generated for each of the sample log types, SendMail, SSH and DHCP, across each of the distance models selected.

Kolmogorov–Smirnov and Kuiper results were not post processed as they gave a direct result as a p-value percentage 0-1. For Kolmogorov–Smirnov and Kuiper results they should also be inverse in comparison to the other distance measurements. They should start off at 1 (100%) as the distributions should be the same, and as the percentage of bad log lines is increased the probability measure should fall as the distributions diverge when compared to all good log lines.

The Quadratic and both Earth Mover's distance results were normalised from 0 to 1 to allow for direct comparison of all the results. It is expected for these results that as the number of known bad log lines are introduced the distance measure should increase when compared to all good log lines.

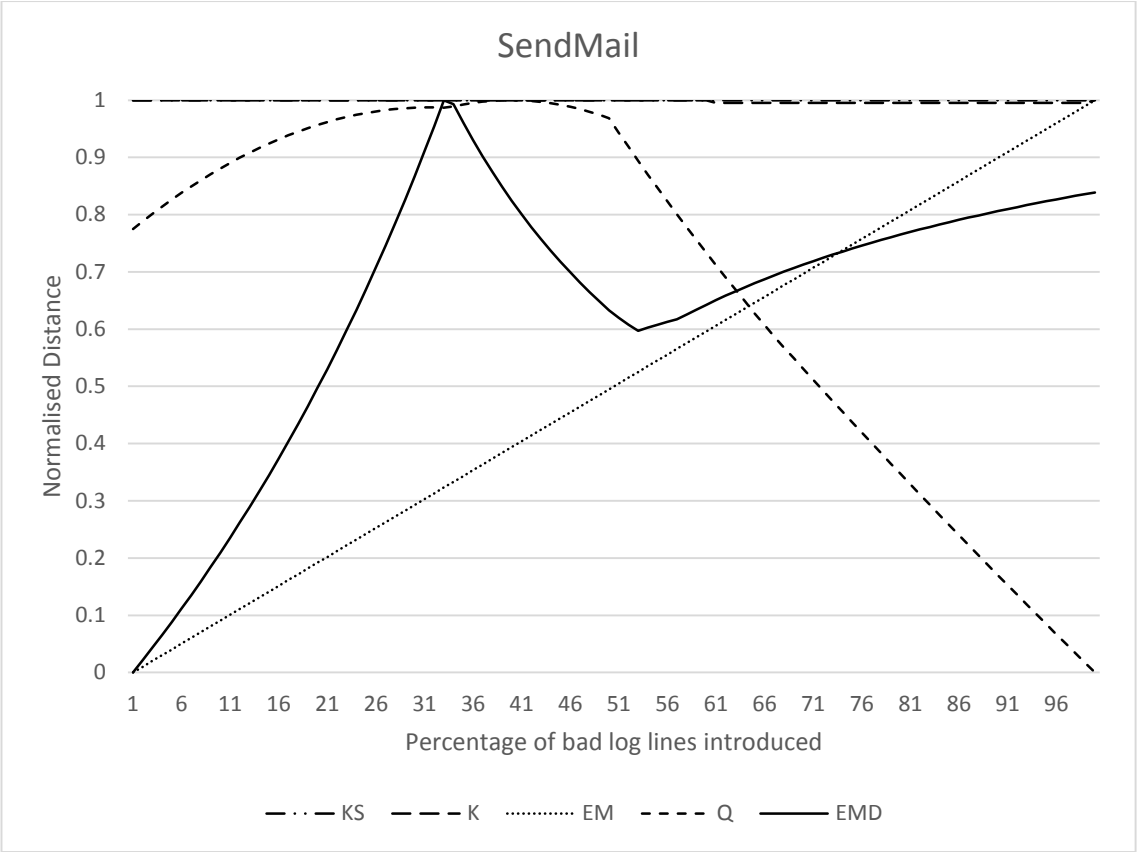


Figure 4-8 SendMail

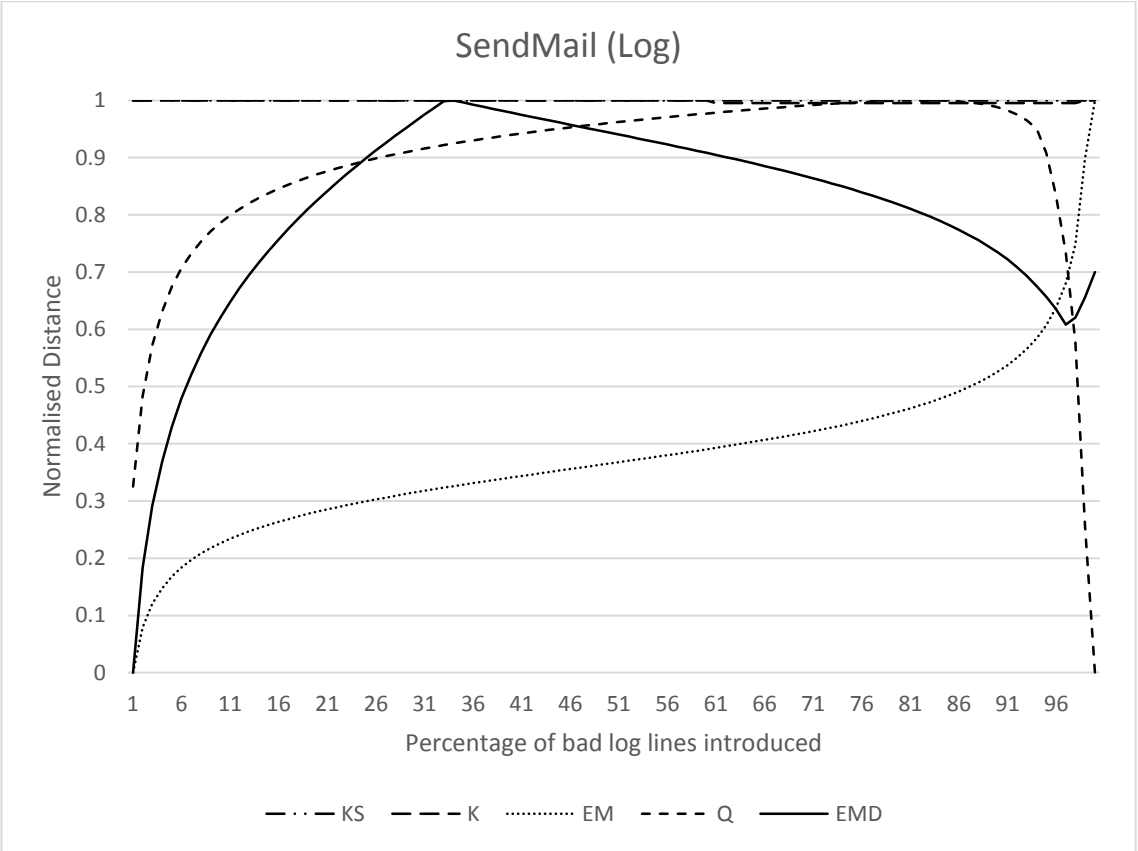


Figure 4-9 SendMail (Log)

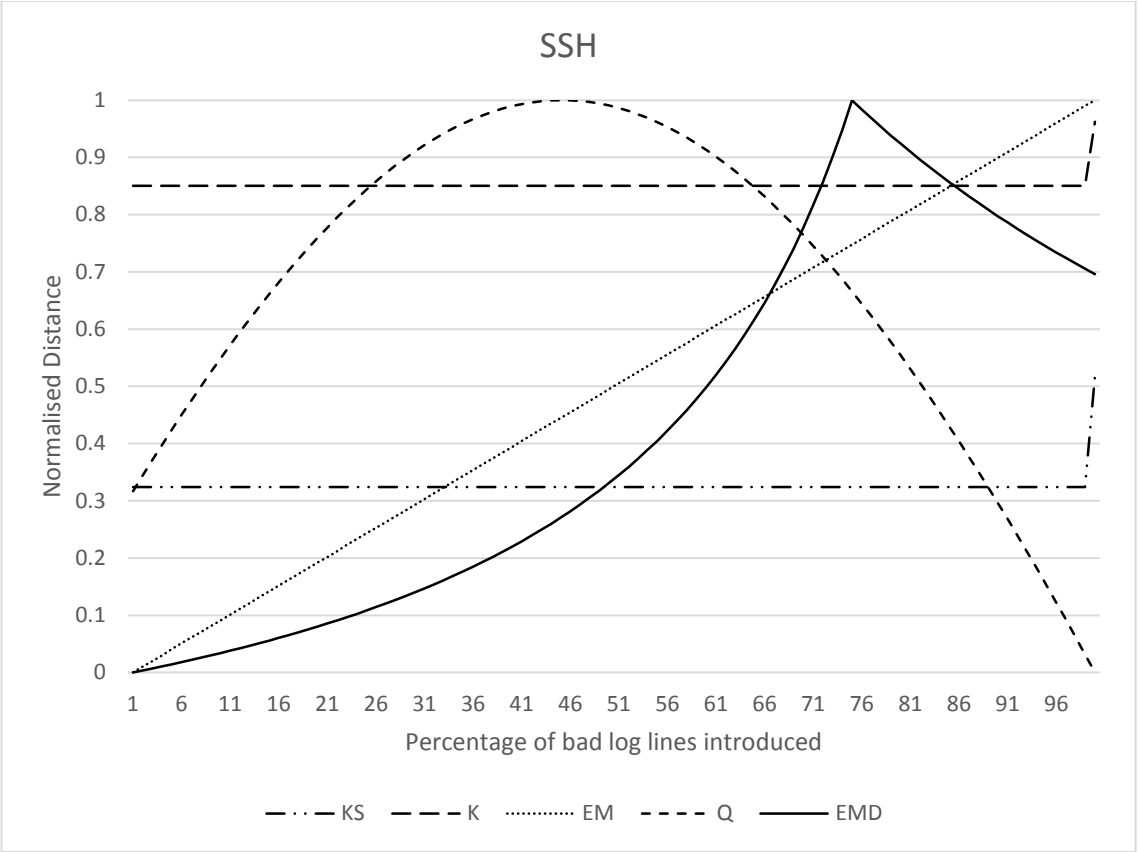


Figure 4-10 SSH

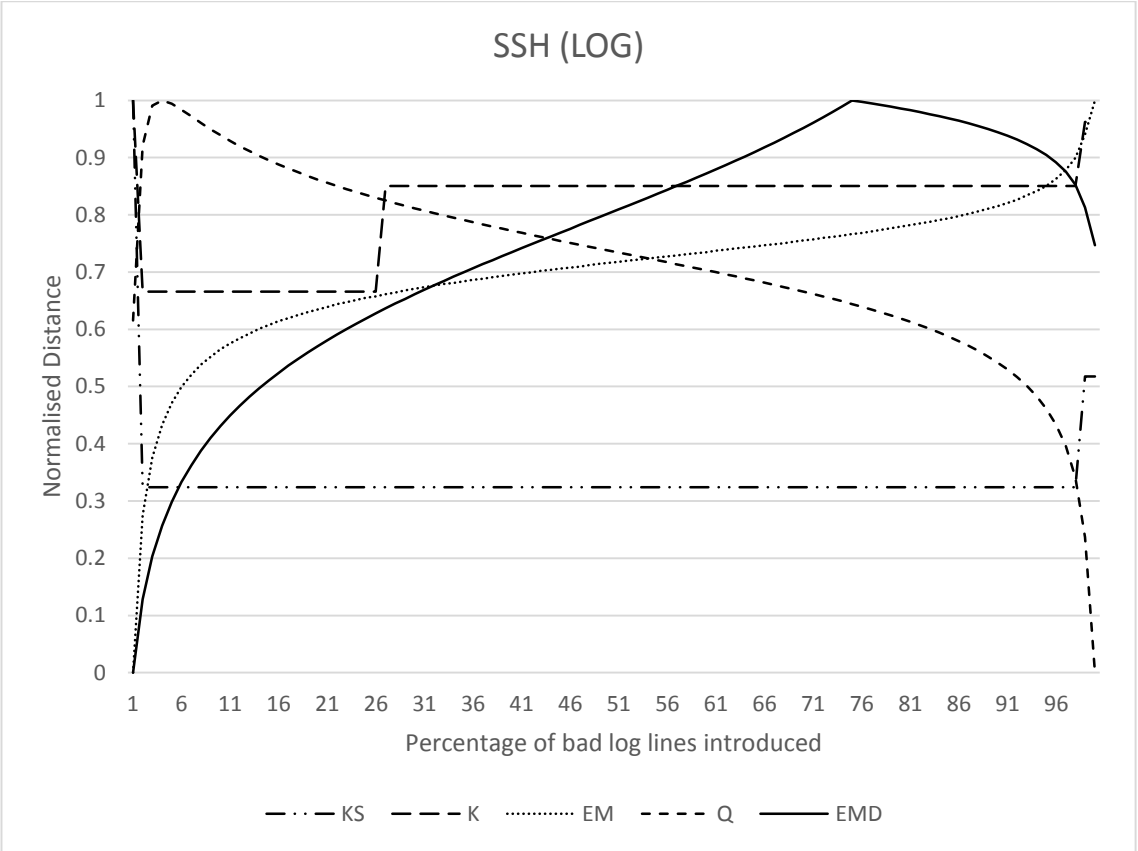


Figure 4-11 SSH (Log)

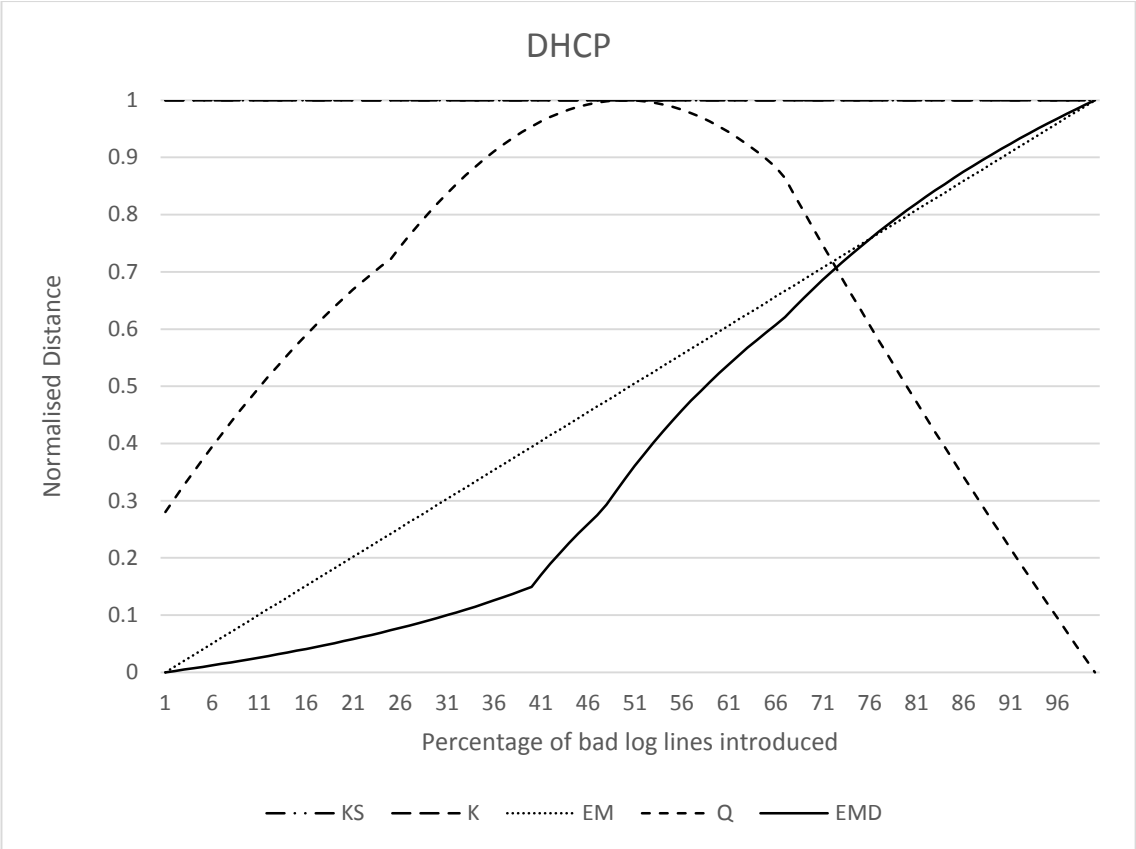


Figure 4-12 DHCP

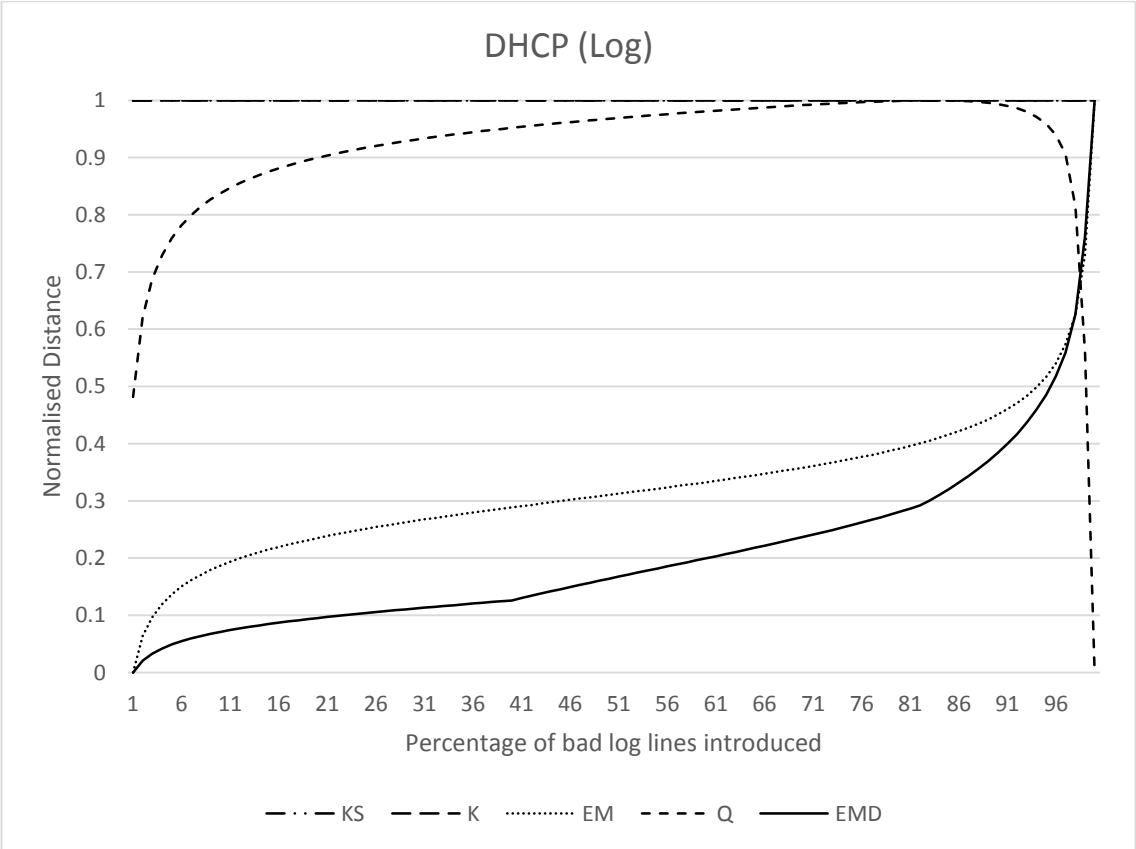


Figure 4-13 DHCP (Log)

Examination of Figure 4-8 through to Figure 4-13 produces a number of interesting observations. In this supervised data set environment both Kolmogorov–Smirnov and Kuiper performed exceptionally poorly, and were double checked in case of any induced errors. However a manual check of a sample of the outputs of these two functions produced the same results.

This could be due to the cumulative distribution function not working very well at all across the data change taking place in the data sets, or the data sets being too small to generate an effective probability measurement as both methods are reliant on these underlying functions to produce a result.

The quadratic distance method had a much better performance, and was especially sensitive to small changes. One result which could indicate unexpected behaviour can be seen in Figure 4-11, the log values of SSH data. In this one case it could be considered to be overly sensitive to the change that took place as can be seen by the rise to almost the limit at only a small percentage of introduced bad log lines and then dropping away as the percentage of bad log lines increased.

Both the quadratic method and Earth Mover's Distance (EMD) – Implementation 2 (One to One) had the same pattern of reaching an inflection point and then the result changing direction or rate. In one case Figure 4-9 SendMail (Log) for EMD there was even two inflection points observed. For the EMD function, as it is a one to one measure of bin data to give the distance metric, the maximum distance may occur before the 100% of injected bad log data point as there could effectively be less "dirt" to "move" to make the histograms equivalent after that point as the sample error rate increases.

These results can be compared against Earth Mover's Distance (EM) – Implementation 1 (weighted by histogram total), where in all non-logarithmic scaled data graphs it produced a linear response in relation to the level of injected bad log data. When used with logarithmically scaled data it offered the desired greater sensitivity at low levels of injected known bad log data and showed no negative inflection points as observed in some of the other distance models.

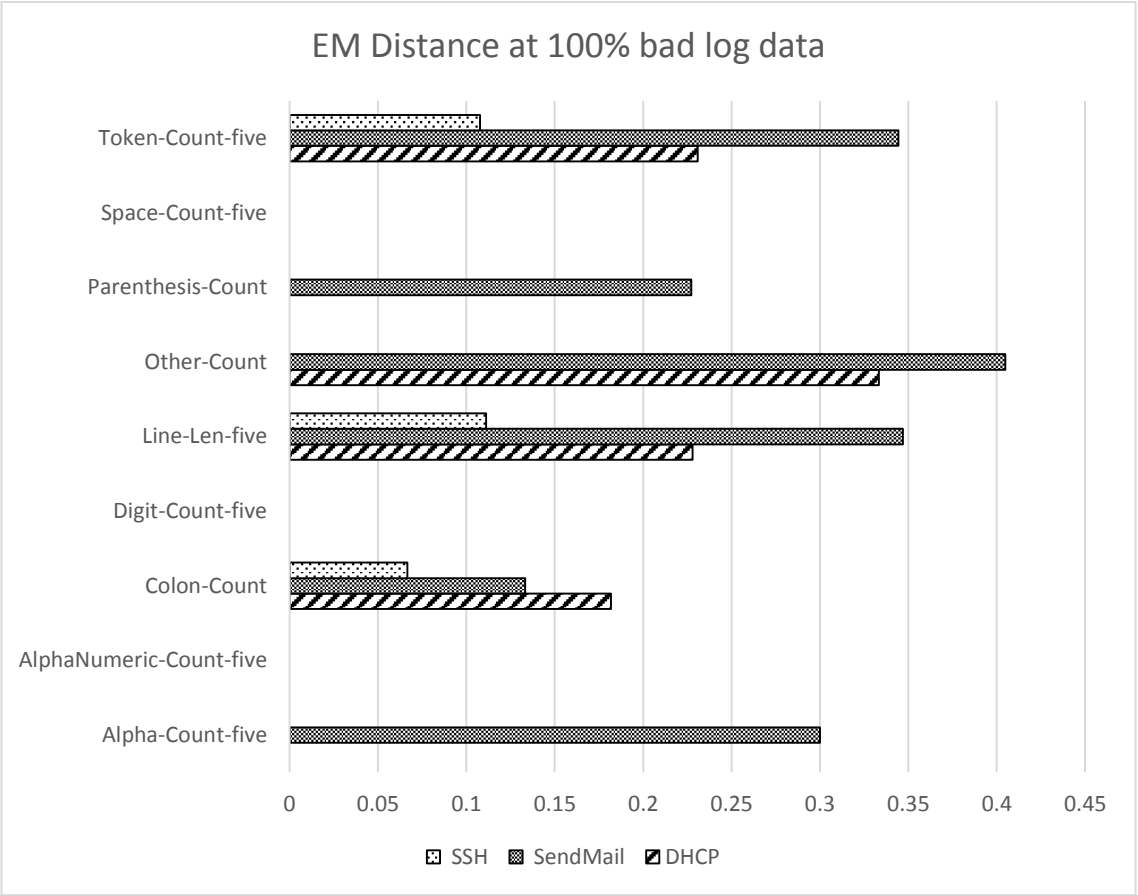


Figure 4-14 EM Distance at 100% bad log data

Given the linear response to the introduction of know bad log data produced by the EM distance model and predictable results across all the supervised data sets, the non-normalised distance results were plotted for each predictor attribute for evaluation as shown in Figure 4-14. When viewing these results we see that across all log file data source types ‘‘space’’, ‘‘digit’’ and ‘‘alpha numeric’’ did not get recorded as they didn’t meet the minimum requirements for the distance model to be generated (histogram length of five). For the other predictor attributes the results were once again predictable and aligned with the expected outcomes. The SSH log file difference between good and bad log data only displayed small spread over three attributes, but was still detectable and measurable. DHCP logs had a greater distance value over four attributes and SendMail logs had the largest distance metric across six attributes.

4.3.4 Summary

Question One – Efficient and accurate predictor attributes

While not the focus of this third iteration of the data mining experiment the results reinforced the findings of iteration one, that the reliance on one log file feature could limit the applicability or sensitivity of the anomaly detection model. As shown in Figure 4-14 it is clear that for different log file types, different predictor attributes can change at different rates.

As raised in iteration two, using the log of the bin size allowed for greater sensitivity. This can be seen clearly when comparing the EM distance plots from Figure 4-8 to 4-13 where at low error rates a much higher distance metric was produced when compared to the linear graph of the non-log results.

For each predictor attribute generated, a histogram has to be built in order for a distance calculation to be performed. One of the requirements set for a distance measurement using the EM distance model was a histogram with a minimum length of five. However, this may have excluded some predictor attributes for certain log file types as shown in Figure 4-14 EM where ‘‘Alpha’’, ‘‘Space’’ and ‘‘Digit’’ recorded no results for SSH, DHCP or Send Mail log file types. This highlights the limitation of relying on any one single attribute as they did in HMAT and in other research where they used word count as the only predictor attribute. The results shown do not mean that these predictor attributes should be removed from further consideration, but rather that in the three log files evaluated they are not performant. However, on other log file types they may be performant.

Question Two – Performant model/s for the detection of anomalies

When comparing the effectiveness of the anomaly detection methods listed as shown in Figure 4-8 to 4-13, Kolmogorov–Smirnov and Kuiper had very poor detection of the introduced known bad data. The Quadratic and Earth Mover’s Distance (EMD) – Implementation 2 (One to One) both had inconsistent results, and as a result, present the risk that they might be interpreted incorrectly. Comparing these models to the Earth Mover’s Distance (EM) – Implementation 1 (weighted by histogram total) shows that it performed the most consistently and accurately across all the anomaly/distance detection methods.

The consistency of the Earth Mover's Distance (EM) – Implementation 1 (weighted by histogram total) results, including not generating a negative slope for the distance measurement and increased sensitivity when using the log values of the histogram bin value, makes it a very effective distance model for use further in this experiment. Its ability to accurately detect the introduction of known bad log lines and thus anomalies based on the change of the log file mix and generate a simple distance metric, allows the EM model to be used as a change/anomaly detection system across a range of predictor attributes.

The Earth Mover's distance can be computationally expensive, but as noted by Rubner et al., 2000 there are now efficient algorithms for the calculation of this distance. Also given the time interval between each result set (one minute), it gives a significant amount of time in order to process the results so this potential expensive calculation could be considered performant based on computational cost.

Using one thousand events for each histogram worked well, and offers possibilities to adapt the framework to slow data rates. For example each time one thousand events are collected the cached histogram data set is updated. The rest of the framework could remain the same with cached results being logged every minute but the data set only being updated when enough events have been collected.

Question Three – Evaluation of anomaly detection model/s.

Given the consistency of the Earth Mover's Distance (EM) – Implementation 1 (weighted by histogram total) model a distance (change) measurement was easy to obtain and interpret. A higher reading always corresponded to a greater relative change in the histograms of the log file attribute being tested.

Sensitivity to smaller changes can be increased by using the log of the bin values, and this process introduced no abnormal effect on the distance measurement and thus the results can still be directly interpreted. A threshold was not established during this iteration, and will be explored in iteration four. No baseline comparison issues were examined in this iteration either, as all results were directly compared with a known "good" sample.

Question Four – Practical and scalable performance constraints.

No issues or performance constraints were uncovered during this iteration, all tools created and used to test this iteration did not encounter any issues that might impact the ability for further development of this work.

4.4 Iteration Four

4.4.1 Business and Data Understanding

The main goal of switching to a supervised data set has been achieved, i.e. the validation of an efficient and effective change detection model to use for anomaly detection. The results show that the EM distance model is by far the best for use in this experiment.

The business objective of this iteration was to apply the knowledge and understanding gained so far, back to the previous unsupervised data sets collected before. This utilised the predictor attributes defined so far, the Earth Mover's Distance (EM) – Implementation 1 distance model as verified in iteration three.

The predictor attributes selected were retested against this much larger data sample to confirm whether they were appropriate for use. The EM distance model was run against all the anomaly predictor attributes. The data was examined using a variety of techniques in order to confirm understanding, and to generate knowledge about how to develop and apply an appropriate anomaly threshold.

As discussed in the literature review, it can be considered that a level of variance or change in the mix of log files can be considered normal. By introducing a threshold governing what is an acceptable level of change occurring in the mix of log file lines an anomaly detection system can finally be developed. Anomalies are detected based on the assumption that when the change in the mix of log file lines based on the predictor attributes generated is larger than the threshold set, the mix of log file features has changed to the point where it could be considered anomalous to the normal pattern of events expected by the system administrator.

Each logging system or service may exhibit different levels of variation in predictor attributes based on what systems and services logs are being collected, it is up to the

system administrator to set and tune this threshold to a level that gives a balance between anomaly detection and unnecessary alerts (false positives).

To determine if it was a false positive it will require a domain expert such as a system administrator to review the change in the mix of events and if required to further examine the actual log files that generated those events with other tools. This will allow the expert to determine if it was a false positive or not and if too many alerts are being generated then it may be required to adjust the detection threshold for the predictor attribute that caused the alert.

This aligns with the original goal of this experiment, which is not to detect a single event but rather to detect when the mix of log files has changed beyond an acceptable level and thus could be considered anomalous. From this detection it can guide an expert or systems administrator to the time and location of the change to enable them to investigate what has changed via other tools.

One threshold test that is commonly used by many IT service providers and monitoring systems is a percentile measurement. Typically this is used in traffic usage type measurements to remove the top 5% of traffic for a burstable billing implementation (Wang, Chen, Yang, & Zheng, 2007). While usage in this way effectively removes the anomalies (traffic spikes) and thus allows the top 5% of traffic not to be billed, the opposite effect is desired in this work.

By implementing a percentile threshold it can be used to identify the top X% of results based on the distance measurement which represents the change in log file mix and thus could be used for the detection of log file anomalies. The percentile threshold measure itself is easy to understand and can be adjusted as required in order to allow for an appropriate level of change (anomaly) detection as described previously.

This fourth iteration completed the data mining cycle, bringing this experiment to a close.

4.4.2 Data Preparation and Modelling

With the EM distance model confirmed via the use of a supervised data set, this iteration returns to the unsupervised real-word data collected over the ninety day period in iteration two. The dataset was extracted from the CouchDB database to be

processed by a testing script. The testing script recorded the appropriate calculated distances using the EM model (current to previous). The logarithmic values were used to increase the sensitivity at the lower histogram bin ranges where small changes might be missed. All datasets were normalised and made equal length prior to calculating the distance.

In this iteration we continued to compare back to the previous sample as a baseline. In addition a sliding window average of the EM distance metric was also created which offered a method for smoothing and setting an anomaly threshold over time based on the EM distance measured and the percentile method discussed in the business and data understanding section.

4.4.3 Evaluation

An R violin plot was generated using the simple 'current to previous' EM distance measurement across all predictor attributes for the entire 126,000 result set to observe the distribution of the EM distance measurements generated and the quartile ranges.

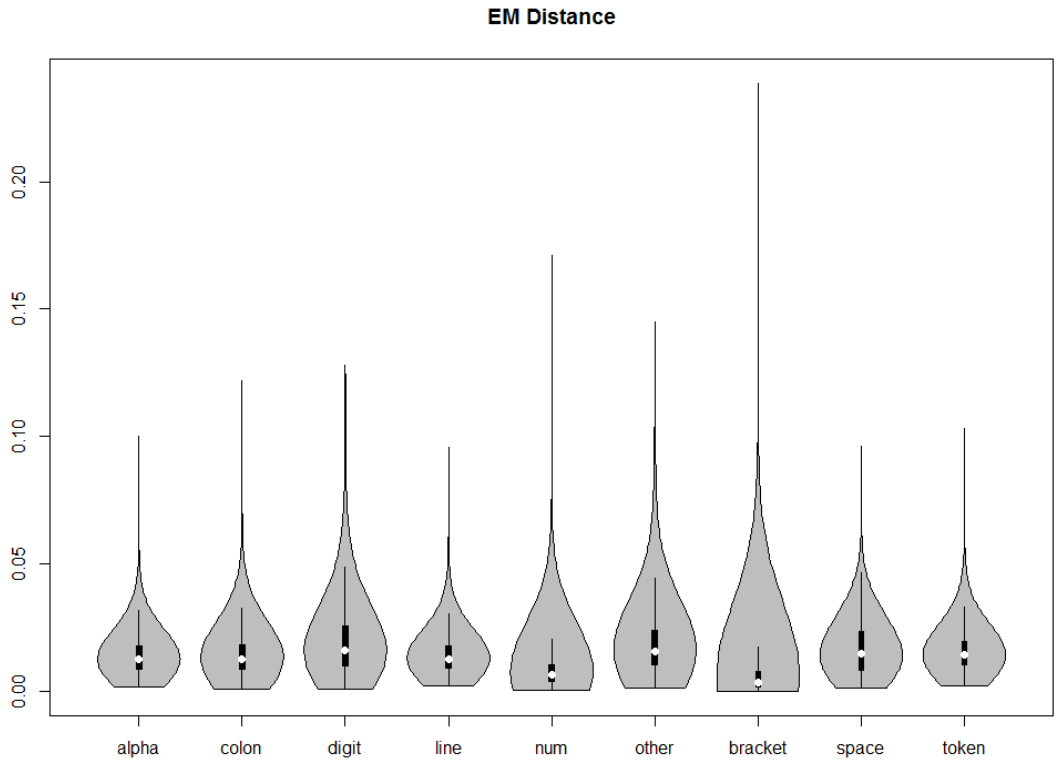


Figure 4-15 EM (Weighted) Violin Plot

From this Violin plot (Adler, 2005) using the default kernel settings and a Tukey outlier definition of 1.5 x inter quartile range (IQR) we can see that there was a range of EM distance measurements generated by comparing current to previous. All predictor attributes showed a range of results as indicated by the box plots with only the predictor attributes ‘‘brackets’’ and ‘‘numbers (Numeric)’’ having a lower quartile range than the others. In addition outliers can be seen as indicated by the peaks of the distribution plots for each attribute. The size of these peaks for each predictor showed a different level of sensitivity to the change in the mix of predictor attributes.

The distribution plots show a skew distribution towards the smaller numbers, this might imply another underlying symmetry to the results which could be used in order to obtain an anomaly measurement.

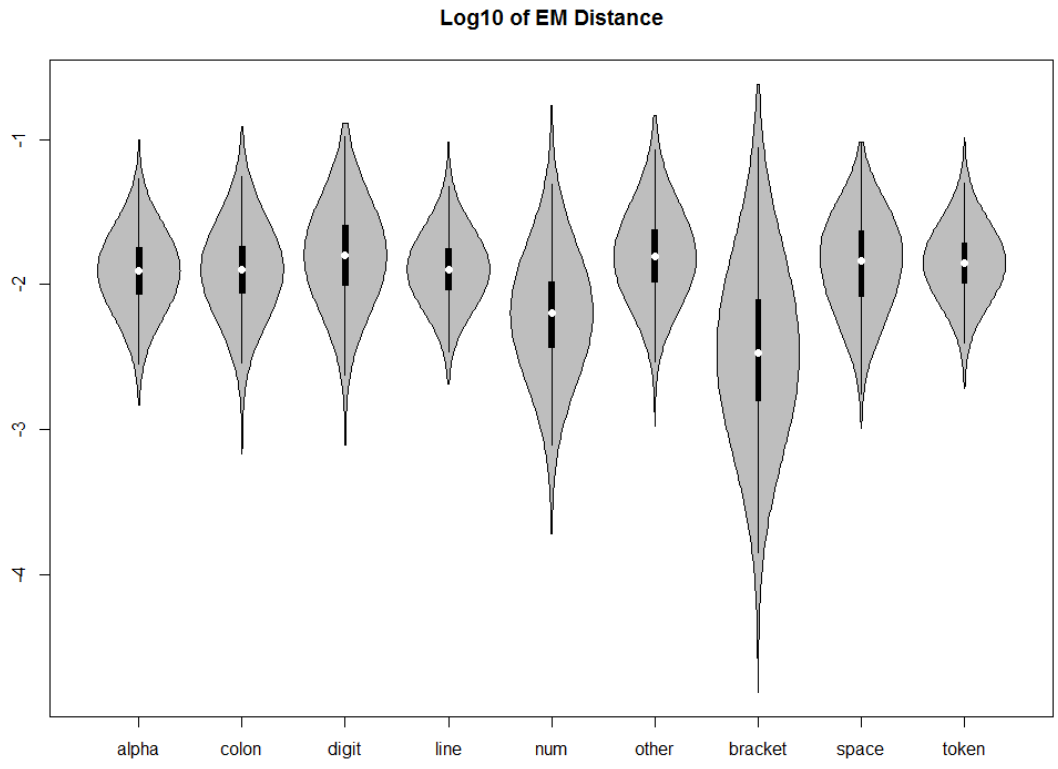


Figure 4-16 EM Log10 (Weighted) Violin Plot

Taking the log of the EM Distance values made the Violin plots more symmetrical which might allow for other methods to evaluate a way to set a threshold of difference to be used. For example HMAT used a weighted standard deviation of an individual

attribute bin in order to determine if a value was abnormal or not. To see if there was any further depth to this result it was decided to investigate this symmetry even further via a default R density plot across all the attributes.

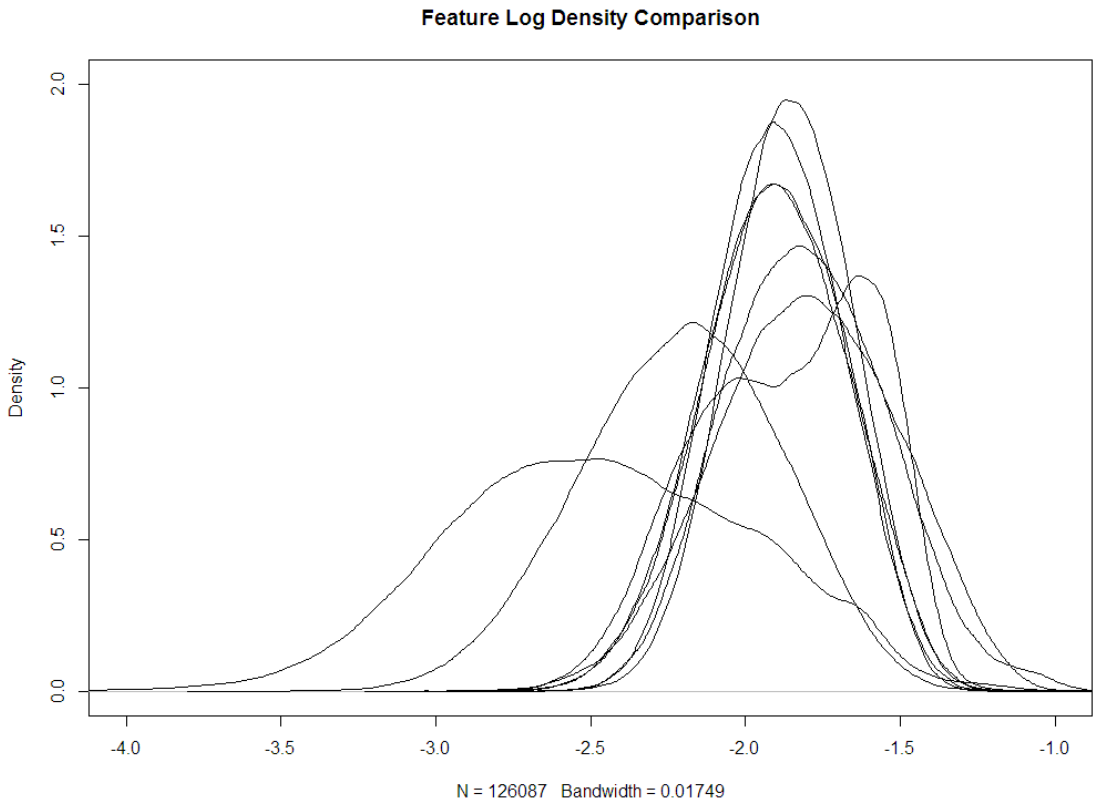


Figure 4-17 EM Feature Log Density Comparison

Closer examination by creating a standard R density plot ("Kernel Density Estimation," 2015) as shown in Figure 4-17, which uses the log value of the earth mover's distance results and overlays each attribute, shows that while some of the attributes approached a normal like distribution, this assumption cannot be held across all attributes and thus further investigation was halted.

As all attributes/features could have relevance in anomaly detection the author decided to focus on log line length again. Given the consistency of the EM distance model the results obtained could be applied back to all the attributes/features as required.

In addition to the focus on log line length, a sliding window average of 10,000 minutes (approximately 7 days) was incorporated at this point to generate a percentile threshold that could be tested against. Any earth mover's distance above a specified percentile threshold was then identified and displayed over an R smoothScatter density plot ("smoothScatter," n.d.) of all the distance results in order to examine the possibility of using a percentile limit as an anomaly detection method.

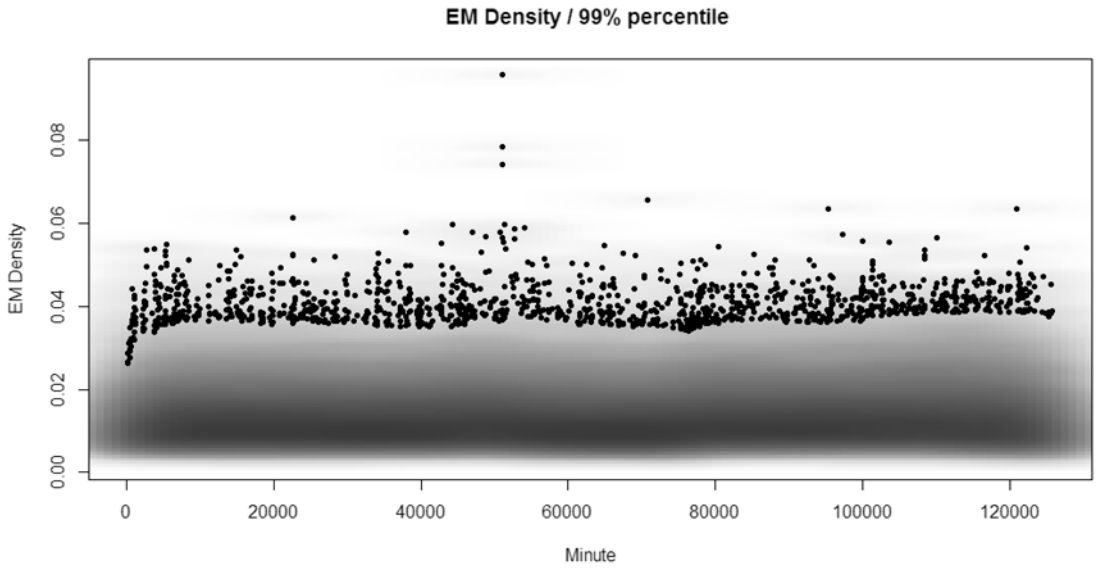


Figure 4-18 EM Density / 99% percentile

As plotted in Figure 4-18, an average of 16.28 anomalies (change in the mix of log file lines) per day were recorded at a 99% percentile limit using a sliding window of 10,000 minutes to generate the EM distance percentage threshold. These results were generated over a total of 126,180 minutes. Given 1440 minutes per day, 1% of events per day would give on average 14.4 anomalies. This result demonstrates a practical level of events that could be investigated further each day.

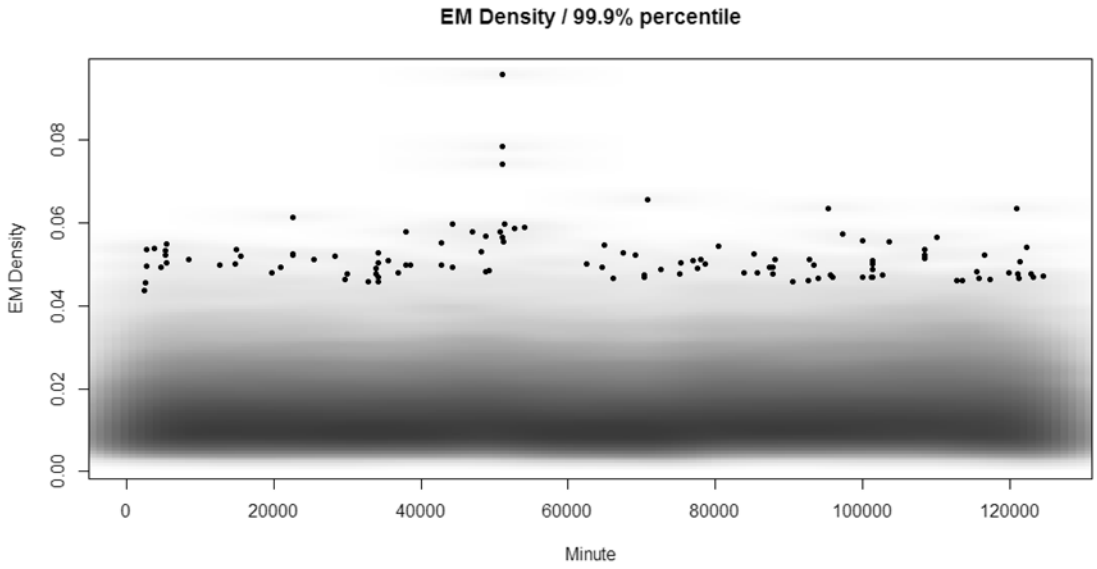


Figure 4-19 EM EM Density / 99.9% percentile

In a second plot, the percentile threshold limit was adjusted to 99.9% something that an end user of the solution could easily do in order to set an anomaly threshold level to one that could be more appropriate for the environment and log file mix. As show in in Figure 4-19, by moving to a 99.9% percentile limit it reduced the event count as expected and gave an average of only 1.3 anomalies per day. This reduction in events also uncovered that there are potentially clusters of events on certain days or over a small range of days which may more strongly indicate an abnormal mix of log file anomalies happening.

A visual examination of a sample of the anomalies being detected was performed. All anomaly events that were examined showed a marked change in the histograms being compared and thus the mix of log file events. This aligns with the results obtained during iteration three of the experiment.

4.4.4 Summary

Question One – Efficient and accurate predictor attributes

For the unsupervised dataset of real-world data, a good range of variance was once again shown across all attributes as shown in Figure 4-15, 4-16 and 4-17. The EM distance generated from these attributes produces outliers which are directly related to the changes in the mix of the log files as demonstrated in "Iteration Three" Figure 4 -14 which gives accurate anomaly detection.

It should be noted though that there is an underlying data bias to the unsupervised data set used to test these attributes. The log files were generated from a range of systems in use at a University, and thus while having a wide range of data types it will likely suffer from a data bias. Different types of organisations, systems and services can exhibit a wide range of log file types along with a range of work patterns causing temporal variance and volume changes.

Question Two – Performant model/s for the detection of anomalies

The EM distance function was used exclusively during this fourth iteration given the consistency of the results obtained during the previous iterations. No unexpected results were identified from this, and thus the author concludes that the EM distance model should continue to be used. Attempts to make the EM distance metric results more symmetrical did not offer any more value to the overall experimental goals or lead to alternative ways for the threshold to be set.

Question Three – Evaluation of anomaly detection model/s.

The evaluation of a percentile limit in order to set a threshold for anomaly detection was the main goal of this fourth iteration. By averaging the EM distance percentile threshold over a sliding window, it created a way to calculate an accepted level of difference/change of distance between the current and previous histogram for the predictor attribute being tested. Anything that then fell outside of this limit of acceptable difference/change was then considered as an anomaly for the purposes of this experiment.

The percentile limit is an easy to understand statistical measure widely used in the IT and service industry that can be easily adjusted to change the number of anomaly events generated to a level acceptable to the end user and data at hand. To extend this across all predictor attributes, each predictor attribute could be individually monitored and have an appropriate threshold set. If one attribute generated too many false alerts its threshold could be adjusted as required. If the attributes instead of being treated as separate were to be combined for a total distance, care would have to be taken due to the difference in variance between attributes. As shown in Figure 4-15, this may supress the smaller range attributes from generating any alerts or impart

an unintentional data bias to the solution. Further work in combining attributes for multivariate anomaly detection is left for future work.

The anomaly threshold method tested was shown to generate appropriate anomaly events based on the changes to the mix of an attribute feature from streaming log files.

Question Four – Practical and scalable performance constraints.

One new feature needed during this fourth iteration was the calculation of a sliding window of the EM distance for each predictor attribute. Its implementation would have to be carefully considered in a full streaming log file solution. Given that there is a minute between each difference calculation this gives ample time for a number of methods to be used.

One method that would align with the current methodology would be leveraging MapReduce functionality for the creation of the sliding windows. Work in this area of has already been undertaken in research by Li, Noorian, Moss, & Leong, 2014 entitled ‘‘Rolling window time series prediction using MapReduce’’ which could be leveraged to produce a practical and scalable solution.

4.5 Data Mining Summary

The time limit for any further data mining iterations was reached, so a complete summary of all results to date will now be presented.

Question One – Efficient and accurate predictor attributes

Although syslog worked well as a message transport, its own features ‘‘Facility’’ and ‘‘Severity’’ were not well implemented by end user systems and thus were removed from the attribute list. By removing these transport specific attributes it also allows for more message transport methods to be used, aligning it with the wide general applicability desired from this work.

Based on the results to date the selected predictor attributes of Alpha Numeric, Alpha, Number (digit), Colon, Parenthesise (Brackets), Space (Tab, Newline, Return), Line Length (Count), Token Count (Word Count) and Other (anything else not already matched) worked well across the real-world log file streaming data set and should be

continued to be used as they broaden the log file changes that can be detected thus increasing the general applicability of the framework to a wider range of log file types.

The one minute sample rate may cause issues for slow data rates. By not having enough log events the histograms generated may have too much variation in them to create and store a stable baseline for comparison. A minimum event limit, for example a one thousand log events as was tested during the supervised distance model testing, performed well. By adding caching to the results it may be possible to deal with slow data rate issues as well.

Each log file data stream generated a range of variance across different predictor attributes. This necessitates a flexible approach in order to reduce any potential data bias in attribute selection. It is therefore recommended that all predictor attributes are recorded and used if they meet the minimum requirements for the EM distance model (number of bins and samples). By using all the attributes, it widens the applicability of the log file types that could be used for anomaly detection, instead of picking on one or a few attributes which may limit the performance of the end solution or introduce an unintentional data bias. This creates a more generalised approach matching the desired outcomes of this experiment.

Another important finding over the course of this experiment was that with the move to a NoSQL (document) storage type, that can support MapReduce type operations, that bin width selection is no longer a critical selection issue as NoSQL is not reliant on a fixed dataset but a flexible document store. From this, bin width has the potential to move to another parameter that could be adjusted by the end user if required to reduce noise in the histogram. Going forward it is recommended that all data should be recorded at the raw value (single bin width value). This will allow the framework to have flexible binning options. Using MapReduce operations affords the ability to adjust bin size as required allowing the framework to support the widest range of log file data structures and types possible.

To manage changes in data volume, histograms should be normalised in order to remove any effect the volume of log files will have on the distance model calculation. If the mix of the histograms remains the same over the sample size then the max bin value should remain relative when compared to all other bin values.

By taking the logarithmic value of the histogram bins before comparing them with a distance model it was found to make the distance model much more sensitive to smaller changes in the low volume attributes. Whereby stopping the larger bin values becoming too dominant in the distance calculations.

The one minute sample time also balances data sample size against the timely detection of anomalies. After discussions with a domain expert it was found that in practice having windows of less than one minute for log file anomaly detection would not offer any further real advantages given the outcomes desired. A one minute window gives a balance between timely notification of anomalies and sample size.

Question Two – Performant model/s for the detection of anomalies

The baseline comparison for the distance model used the current histogram compared to the previous histogram. By doing this an immediate change in the mix of log file attributes can be detected, but a slow change over a number of samples may be missed. This simple approach offers the broadest generic approach to this problem area, but does have some limitations as noted.

These limitations could be overcome by implementing other baseline models which create averages, or weighted models of attribute histograms for comparison to the current histogram, but this will be left for future work. The methods used so far in this experiment will not preclude any of these alternative baseline calculation types from being used.

A major finding from this experiment was the EM (Weighted) earth mover's distance model and how well it performed with the supervised data set against the log file attributes being tested. It gave consistent results that are easily measurable and can be directly interpreted. All the other models tested had a number of short comings which would make their use in this field impractical. The one down side to this model is that it is reasonably computationally intensive. However, the Earth Mover's distance model is a topic of current research and its performance continues to be improved on as Huang, Zhang, Buyya, & Chen, 2014 have shown in their work "MELODY-JOIN: Efficient Earth Mover's Distance similarity joins using MapReduce". Their use of MapReduce with Earth Mover's matches with the MapReduce paradigm that has aligned very well in this data mining experiment to date.

Question Three – Evaluation of anomaly detection model/s.

In order to set a threshold of when something would be classed as abnormal or ‘‘too’’ different one further baseline was needed to be established. The method used was a sliding window of the average percentile limit for the calculated EM distance for a log file predictor attribute sample histogram comparison. For the purposes of this experiment a sliding window of ten thousand records (minutes) was used which is approximately one week (one week is a common temporal range for many systems and services which have a human element.) This window will need the ability to be adjusted as required in order to support other temporal ranges based on the temporal pattern of the streaming log files being processed.

Any distance measure above this percentile EM distance threshold was then considered an anomaly. This anomaly threshold is an easy to understand metric which can easily be tuned by a system expert as required to meet the needs of the user and the streaming log files at hand. When used in this way, the combination of methods and attributes was shown to generate reliable anomaly events based on the changes to the mix of an attribute from streaming log files.

Each predictor attribute can be individually monitored (or ignored) and an appropriate threshold set for each one. As noted this is easily adjusted so for a particular environment if one attribute is generating too many false alerts its threshold could be adjusted as required. The limitation here is that this decision will need to be carried out by a system expert experienced in the examination of log files. Over time the framework will become tuned to the individual anomaly detection levels required.

Question Four – Practical and scalable performance constraints.

The ultimate outcome of this work was the identification of a practical and scalable way for the detection of anomalies in streaming log files. A common finding across much of this research was the need for a high level of flexibility in order for it to be adapted to different streaming log files rates, file types and temporal patterns.

In addressing the practical aspects of this experiment, the proposed framework has the ability to support many log file transport types, with syslog as common default that could be used without it being restricted to this format. Commonly available tools and

resources were used, with no restrictive licencing practices and thus offering an easy way to implement and extend this work in the same way if so desired. Flexibility is offered across all the key features including bin sizes, baselines (windows), log event rates, sliding windows for change detection, and the setting of easy to understand thresholds, thus allowing the broadest general applicability.

The only real restriction is that this work has been designed and tested with only a single event per log line, multiline log files were not tested, but given the generalised nature of the approach this method should still be applicable if each line contains enough variance in attribute features (length) or multiline log events could be consolidated into a single message by pre-processing.

To ensure that the framework was scalable, it was found that leveraging the combination of NoSQL and MapReduce programing paradigms allows the ability to not only scale this framework easily, but to also directly support the required data mining processes (Sliding window, Aggregation, Earth Mover and Visualisation etc.) The sample rate of once a minute also allows a range of opportunities for further aggregating the data as required and provides a practical time period in which to perform the anomaly check.

By utilising NoSQL storage it also provides the ability to store all the log file events as well given the huge scalability of the applications in this space. If an anomaly is detected it would allow for the user to examine in depth the actual log events that caused it. Even if these full logs could only be kept for a short period of time (a few days) the actual predictor attribute data could be kept for a much longer timeframe as it is only a relatively small amount of data being recorded every minute. This presents the possibly for long term, or seasonal baseline generation of predictor attributes to compare against for the distance model.

Overall, this approach aligns well with a visualization methodology generated by Shneiderman, 1996 which is summed up in the following quote, “Overview First, Zoom and Filter, Details-on-Demand“. This work effectively supports the first and second parts of this visualisation methodology, the “Overview” and then identification of where to “Zoom” based on when the anomaly is located. This allows for other tools to

be used in this space which are not capable of working with streaming log files or large data volumes by narrowing the data set to investigate the anomaly further.

4.6 Proposed Framework

This flow diagram shows the proposed framework as outlined in the data mining summary. The framework is intended to be implemented using NoSQL and MapReduce as core design methods.

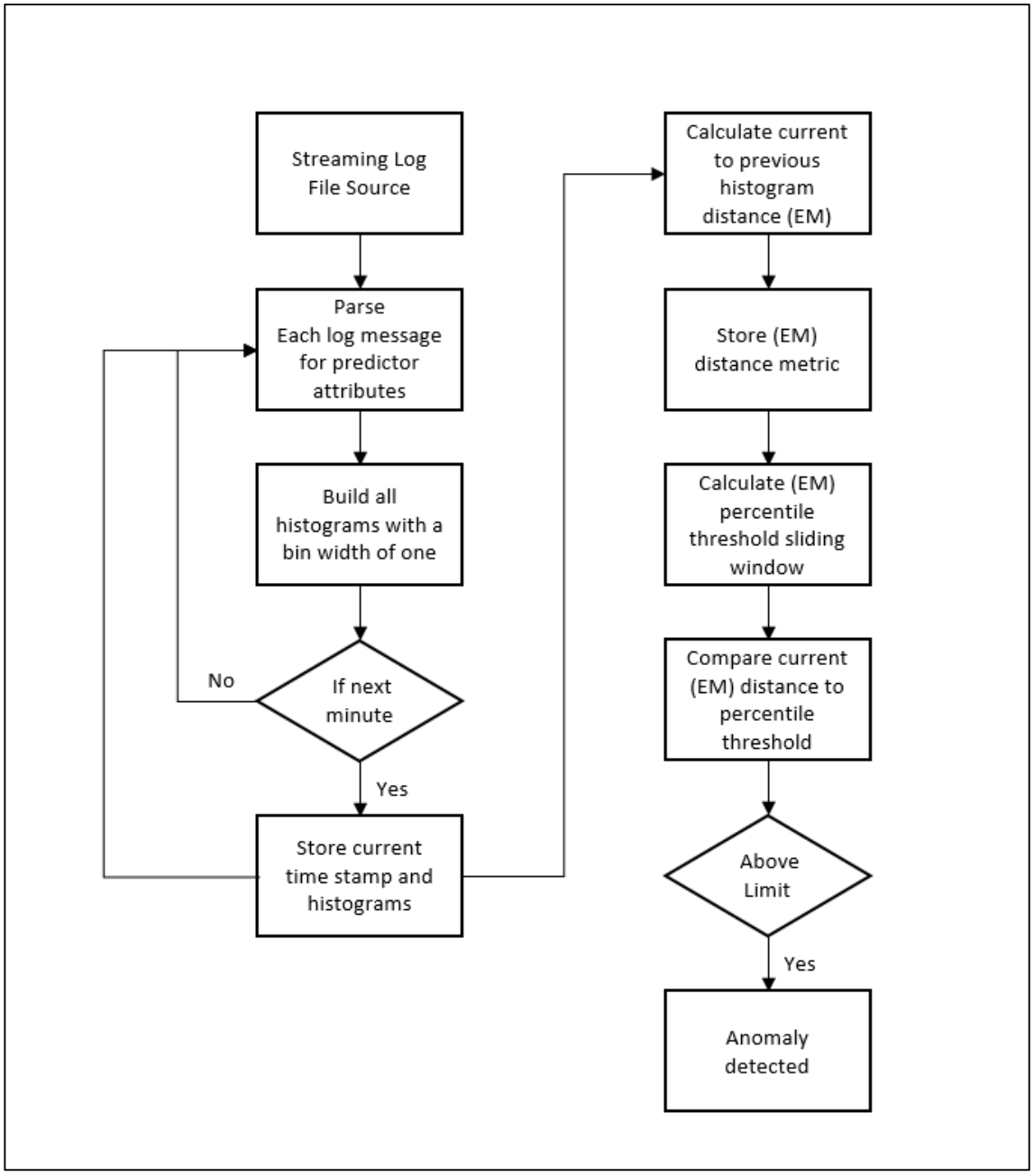


Figure 4-20 Proposed Framework

The framework itself could be developed in a number of ways. For this experiment Perl was the main tool used to build the data mining and processing applications as exact requirements were unknown. By using this method the author retained the flexibility to change the approach as knowledge was gained about the data.

Now that the major requirements are known, there are a number of ways in which this could be developed further. Here are two proposed methods.

Framework one:

Using the requirements and approaches developed as part of this thesis, leverage the log collection and processing pipeline as presented in Chukwa which is now a part of the Hadoop ecosystem ("Welcome to Apache™ Hadoop®!," n.d.).

Leverage the rest of the Hadoop ecosystem to process the data as presented in the flow diagram above.

Hadoop is more suited to non-interactive data mining approaches, but the framework could be designed in such a way as to minimise this possible limitation.

Framework two:

Using the requirements and approaches developed as part of this thesis, combine a number of applications to achieve the overall framework.

Logstash ("Logstash," n.d.) For log transport and processing.

NoSQL Database (To be determined for example CouchDB ("Apache CouchDB," n.d.) or MongoDB ("MongoDB," n.d.)) for anomaly processing

Elasticsearch ("Elastic · Revealing Insights from Data (Formerly Elasticsearch)," n.d.) and Kibana ("Kibana," n.d.) for full text indexing and visualisation.

This framework offer the broadest support for the outcomes of this project and would be the author's preference moving forward.

For each framework, it should have the ability to store every attribute histogram in a NoSQL document, every minute. It should also offer distributed processing and MapReduce on customisable time slices, and bin widths with the ability to compare differences (EM distances) across various sliding time windows.

Chapter 5 - Conclusion

This thesis presented a new and novel approach to the detection of anomalies in streaming log files. This new approach offers a generic, practical and scalable method to achieve this outcome along with offering new insights into this problem area. All requirements to build this streaming log file detection system have been evaluated and tested and a proposed framework for the detection of anomalies in streaming log files has been presented.

The predictor attributes identified and refined from previous research were performant for the anomaly detection methodology used and allow for a variety of log file formats and transport methods to be used if required. Through the use of NoSQL and MapReduce design paradigms the framework has the capability to be scaled as large as required.

The EM distance model used for anomaly detection showed extremely consistent results over a supervised dataset and when applied to real world streaming data gave an effective method of setting a percentile threshold in order to identify when the mix of log file attributes were outside the accepted level of ‘‘change/difference’’.

The framework also supports and aligns with a common visualization methodology for exploring large data sets as proposed by Shneiderman, 1996., defined as “Overview First, Zoom and Filter, Details-on-Demand“. Effectively this anomaly detection solution helps the user know from an “Overview” where to “Zoom” in on the data. From this point the user may apply a “Filter” through the use of further tools or techniques to a smaller data set or time slice to obtain the “Details-on-Demand.”

Chapter 6 - Directions for Future Research

One possible direction for future research is that, given the generic nature of this approach, it could be applied to lines of message text outside of the domain of log files which have a regular structure to them. The predictor attributes selected might have to be adjusted to better suit the domain but the remaining methodology for anomaly detection could remain the same.

Another area that could be developed further or has the capability to improve the anomaly detection process is to extend the baseline options for the histogram comparison method. As already referenced in this experiment, instead of the present method of comparing the current to the previous histogram, there are other baseline methods which may be worthy of evaluation such as a sliding average, exponential, or even a seasonal model such as Holt-Winters (Winters, 1960). Given the flexible and general applicability desired, more than one type of baseline method could be implemented. As the histograms are stored, the framework is not limited to any one baseline model, a variety of them could be calculated by leveraging the NoSQL document store and MapReduce functionality for direct comparison each minute, offering the ability to detect both long term change and short term events.

It would also be interesting to investigate the possibility of combining the univariate approach currently for each attribute into a multivariate approach. Methods to investigate this could include weighting the univariate distance results by histogram length or some other weighting method so the distance result represents a balanced multivariate combination of all the applicable predictor attributes that were generated as a distance measurement.

The final piece of future work envisaged would be the creation of the entire solution so it could be put into production and used to detect anomalies as described in section 4.6, the proposed framework. From this foundation further research could be conducted.

Chapter 7 - References

aarchiba/kuiper. (2009). Retrieved August 29, 2015, from https://github.com/aarchiba/kuiper

Adler, D. (2005, October 29). Violin plot. CRAN. Retrieved from https://cran.r-project.org/web/packages/vioplot/vioplot.pdf

Apache CouchDB. (n.d.). Retrieved from http://couchdb.apache.org/

Butler, J. M. (2009, February). Benchmarking Security Information Event Management (SIEM). SANS. Retrieved from https://www.sans.org/reading-room/whitepapers/analyst/benchmarking-security-information-event-management-siem-34755

Cardo, N. P. (2011). Logjam: A scalable unified log file archiver (pp. 1 –9). Presented at the High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, *41*(3), 15:1–15:58. http://doi.org/10.1145/1541880.1541882

Common Event Expression: CEE, A Standard Log Language for Event Interoperability in Electronic Systems. (2014, November 28). Retrieved September 11, 2010, from http://cee.mitre.org/

EarthMoversDistance (Apache Commons Math 3.6-SNAPSHOT API). (n.d.). Retrieved from https://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/ml/distance/EarthMoversDistance.html#EarthMoversDistance()

Elastic · Revealing Insights from Data (Formerly Elasticsearch). (n.d.). Retrieved July 27, 2015, from https://www.elastic.co/

Fayyad, U., Piatetsky-shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, *17*, 37–54.

Frei, A., & Rennhard, M. (2008). Histogram Matrix: Log File Visualization for Anomaly Detection (pp. 610–617). Presented at the Availability, Reliability and Security, 2008. ARES 08. Third International Conference on. Retrieved from 10.1109/ARES.2008.148

Fu, Q., Lou, J.-G., Wang, Y., & Li, J. (2009). Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis (pp. 149 –158). Presented at the Ninth IEEE International Conference on Data Mining, 2009. ICDM '09. http://doi.org/10.1109/ICDM.2009.60

Gerhards, R. (2009, March). The Syslog Protocol. Retrieved August 14, 2012, from http://tools.ietf.org/html/rfc5424

Hafner, J., Sawhney, H. S., Equitz, W., Flickner, M., & Niblack, W. (1995). Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(7), 729–736. http://doi.org/10.1109/34.391417

Howard, L. (n.d.). Net::Syslog - search.cpan.org. Retrieved from http://search.cpan.org/perldoc?Net%3A%3ASyslog

HP ArcSight Connectors - HP Enterprise Security. (2012, October 5). Retrieved October 5, 2012, from http://www.hpenterprisesecurity.com/products/hp-arcsight-security-intelligence/hp-arcsight-connectors/

Huang, J., Zhang, R., Buyya, R., & Chen, J. (2014). MELODY-JOIN: Efficient Earth Mover's Distance similarity joins using MapReduce. In *2014 IEEE 30th International*

Conference on Data Engineering (ICDE) (pp. 808–819). http://doi.org/10.1109/ICDE.2014.6816702

Kernel Density Estimation. (2015). Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/stats/html/density.html

Kholghi, M., Hassanzadeh, H., & Keyvanpour, M. (2010). Classification and evaluation of data mining techniques for data stream requirements (Vol. 1, pp. 474 –478). Presented at the 2010 International Symposium on Computer Communication Control and Automation (3CA). http://doi.org/10.1109/3CA.2010.5533759

Kibana: Explore, Visualize, Discover Data | Elastic. (n.d.). Retrieved August 1, 2015, from https://www.elastic.co/products/kibana

Kolmogorov-Smirnov and Kuiper's Tests of Time Variability. (2012, December 14). Retrieved from http://cxc.harvard.edu/csc/why/ks_test.html

Kurgan, L. A., & Musilek, P. (2006). A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review*, *21*(01), 1–24. http://doi.org/10.1017/S0269888906000737

Li, L., Noorian, F., Moss, D. J. M., & Leong, P. H. W. (2014). Rolling window time series prediction using MapReduce. In *2014 IEEE 15th International Conference on Information Reuse and Integration (IRI)* (pp. 757–764). http://doi.org/10.1109/IRI.2014.7051965

Liu, Y., Pan, W., Cao, N., & Qiao, G. (2010). System anomaly detection in distributed systems through MapReduce-Based log analysis (Vol. 6, pp. V6–410 –V6–413). Presented at the 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). http://doi.org/10.1109/ICACTE.2010.5579173

Logstash: Collect, Parse, Transform Logs | Elastic. (n.d.). Retrieved August 1, 2015, from https://www.elastic.co/products/logstash

Makanju, A., Zincir-Heywood, A., & Milios, E. (2012). A Lightweight Algorithm for Message Type Extraction in System Application Logs. *IEEE Transactions on Knowledge and Data Engineering*, *PP*(99), 1. http://doi.org/10.1109/TKDE.2011.138

McQueen, T. (n.d.). Algorithm::Diff - search.cpan.org. Retrieved from http://search.cpan.org/perldoc?Algorithm%3A%3ADiff

Mohammadjafar Esmaeili, & Arwa Almadan. (2011). Stream Data Mining and Anomaly Detection. *International Journal of Computer Applications*, pp. 38–41. Bangalore.

MongoDB. (n.d.). Retrieved from https://www.mongodb.com/

MySQL :: The world's most popular open source database. (n.d.). Retrieved from https://www.mysql.com/

Paltani, S. (2004). Searching for periods in X-ray observations using Kuiper's test. Application to the ROSAT PSPC archive. *Astronomy and Astrophysics*, *420*(2), 789–797. http://doi.org/10.1051/0004-6361:20034220

Piatetsky, G. (2014, October 28). CRISP-DM, still the top methodology for analytics, data mining, or data science projects. Retrieved from http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html

Poe, C. (n.d.). HOP::Lexer - search.cpan.org. Retrieved from http://search.cpan.org/perldoc?HOP%3A%3ALexer

Processing.js. (n.d.). Retrieved from http://processingjs.org/

Rabkin, A., & Katz, R. (2010). *Chukwa: A system for reliable large-scale log collection* (No. UCB/EECS-2010-25).

R Core Team. (2015). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, *40*(2), 99–121. http://doi.org/10.1023/A:1026543900054

sawmill.net - Sawmill - Universal log file analysis and reporting. (2012, October 5). Retrieved October 5, 2012, from http://www.sawmill.net/

Sharma, S., & Osei-Bryson, K.-M. (2010). Toward an integrated knowledge discovery and data mining process model. *The Knowledge Engineering Review*, *25*(01), 49–67. http://doi.org/10.1017/S0269888909990361

Sharma, S., Osei-Bryson, K.-M., & Kasper, G. M. (2012). Evaluation of an integrated Knowledge Discovery and Data Mining process model. *Expert Systems with Applications*, *39*(13), 11335–11348. http://doi.org/10.1016/j.eswa.2012.02.044

Shearer, C. (2000). The CRISP-DM methodology: The new blueprint for data mining. *Journal of Data Warehousing*, *5*(4), 13–22.

Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations (pp. 336–343). Presented at the Visual Languages, 1996. Proceedings., IEEE Symposium on. Retrieved from 10.1109/VL.1996.545307

smoothScatter. (n.d.). Retrieved from https://svn.r-project.org/R/trunk/src/library/graphics/R/smoothScatter.R

Splunk | IT Search for Log Management, Operations, Security and Compliance. (2010, September 7). Retrieved September 7, 2010, from http://www.splunk.com/

SQLite Home Page. (n.d.). Retrieved from https://www.sqlite.org/

stat_analysis.pl. (n.d.). Retrieved from http://ftp.filesystems.org/osprof/analysis/stat_analysis.pl

Taerat, N., Brandt, J., Gentile, A., Wong, M., & Leangsuksun, C. (2011). Baler: deterministic, lossless log message clustering tool. *Computer Science - Research and Development*, *26*(3-4), 285–295. http://doi.org/10.1007/s00450-011-0155-3

TMath:KolmogorovTest. (n.d.). Retrieved from https://root.cern.ch/root/html/TMath.html#TMath:KolmogorovTest

Vaarandi, R. (2003). A data clustering algorithm for mining patterns from event logs (pp. 119–126). Presented at the IP Operations and Management, 2003. (IPOM 2003). 3rd IEEE Workshop on.

Vaarandi, R. (2004). A breadth-first algorithm for mining frequent patterns from event logs (pp. 293–308).

V Jyothsna, V V Rama Prasad, & K Munivara Prasad. (2011). A Review of Anomaly based Intrusion Detection Systems. *International Journal of Computer Applications*, pp. 26–35. Bangalore.

Wang, J., Chen, J., Yang, M., & Zheng, S. Q. (2007). Traffic regulation with single- and dual-homed ISPs under a percentile-based pricing policy. *Journal of Combinatorial Optimization*, *17*(3), 247–273. http://doi.org/10.1007/s10878-007-9111-3

Welcome to Apache™ Hadoop®! (n.d.). Retrieved from https://hadoop.apache.org/

Winters, P. R. (1960). Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, *6*(3), 324–342.
Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining (pp. 29–39).

Chapter 8 - Appendices

Appendix A – Data sets used in Iteration three

SSH - Good

Oct 13 09:39:50 server12 sshd[24067]: Accepted publickey for root from 123.45.11.5 port 51915 ssh2

Oct 28 03:00:59 server12 sshd[5496]: Accepted password for vinvinv from 123.45.42.236 port 10673 ssh2

Oct 27 03:00:55 server12 sshd[7666]: Accepted password for vinvinv from 123.45.42.236 port 5752 ssh2

Oct 27 03:00:59 server12 sshd[7709]: Accepted password for vinvinv from 123.45.42.236 port 5753 ssh2

Oct 13 23:53:35 server12 sshd[32002]: Received disconnect from 123.45.11.5: 11: disconnected by user

Oct 13 23:53:35 server12 sshd[32002]: Received disconnect from 123.45.11.5: 11: disconnected by user

Oct 14 01:13:08 server12 sshd[24067]: Received disconnect from 123.45.11.5: 11: disconnected by user

Oct 13 09:39:50 server12 sshd[24067]: Accepted publickey for root from 123.45.11.5 port 51915 ssh2

Mar 19 15:23:58 server12 sshd[15751]: Accepted publickey for jdoeoe from 123.45.11.5 port 38637 ssh2

Oct 13 09:54:37 server12 sshd[32002]: Accepted publickey for root from 123.45.11.5 port 52918 ssh2

SSH-BAD

Oct 14 21:31:43 server12 sshd[24099]: Invalid user from 123.45.60.67

Oct 14 21:31:44 server12 sshd[24118]: Invalid user monitor from 123.45.60.67

Oct 14 21:31:43 server12 sshd[24099]: input_userauth_request: invalid user [preauth]

Oct 14 21:31:43 server12 sshd[24101]: Protocol major versions differ for 123.45.60.67: SSH-2.0-OpenSSH_5.9 vs. SSH-1.33-OpenSSH_5.0

Oct 14 21:31:44 server12 sshd[24102]: Failed password for invalid user manage from 123.45.60.67 port 35087 ssh2

Oct 14 21:31:44 server12 sshd[24116]: Failed password for invalid user admin from 123.45.60.67 port 35102 ssh2

Oct 14 21:31:44 server12 sshd[24121]: Failed password for ftp from 123.45.60.67 port 35104 ssh2

Oct 14 21:31:44 server12 sshd[24121]: Failed password for ftp from 123.45.60.67 port 35104 ssh2

Oct 15 03:00:55 server12 sshd[17408]: Connection closed by 123.45.42.236 [preauth]

Oct 14 21:31:48 server12 sshd[24073]: Did not receive identification string from 123.45.60.67

DHCP – Good

Jul 10 08:53:42 zz-mail dhcpd: DHCPREQUEST for 10.251.17.16 from d4:6d:50:be:62:50 via bond0.459

Jul 10 08:53:42 zz-mail dhcpd: DHCPACK on 10.251.17.16 to d4:6d:50:be:62:50 via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPREQUEST for 10.1.60.57 from 2c:3e:cf:7b:f7:8e via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPACK on 10.1.60.57 to 2c:3e:cf:7b:f7:8e via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPREQUEST for 10.1.60.64 from 2c:3e:cf:7b:f2:d5 via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPACK on 10.1.60.64 to 2c:3e:cf:7b:f2:d5 via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPREQUEST for 10.249.96.37 from 00:10:7f:59:5c:e5 via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPACK on 10.249.96.37 to 00:10:7f:59:5c:e5 via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPREQUEST for 123.45.198.215 from 00:10:7f:43:88:ba via bond0.459

Jul 10 08:53:43 zz-mail dhcpd: DHCPACK on 123.45.198.215 to 00:10:7f:43:88:ba via bond0.459

DHCP – Bad

Jul 10 08:53:42 zz-mail dhcpd: DHCPDISCOVER from 00:13:c3:e3:a0:b2 via 10.1.11.1: network 10.1.11.0/24: no free leases

Jul 10 08:53:42 zz-mail dhcpd: DHCPDISCOVER from 00:15:17:17:e5:56 via 123.45.65.1: network 123.45.65.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 78:ab:bb:ac:68:88 via 10.254.20.1: network 10.254.20.0/24: no free leases

Jul 10 08:53:45 zz-mail dhcpd: DHCPDISCOVER from 00:13:c4:01:d4:34 via 10.1.31.1: network 10.1.31.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 00:1f:67:bf:78:80 via 10.249.96.1: network 10.249.96.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 34:bd:c8:72:9e:f0 via 10.249.97.1: network 10.249.97.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 00:13:c3:9b:0b:f1 via 10.1.67.1: network 10.1.67.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 78:ab:bb:3a:67:7b via 10.254.15.1: network 10.254.15.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 00:13:c4:01:d4:ab via 10.1.61.1: network 10.1.61.0/24: no free leases

Jul 10 08:53:44 zz-mail dhcpd: DHCPDISCOVER from 78:ab:bb:ac:68:86 via 10.254.20.1: network 10.254.20.0/24: no free leases

SendMail – Good

Jun 15 23:43:06 taag sendmail[4104]: s5FBh6qF004104: from=<croncroncron@abc.ac.nz>, size=916, class=0, nrcpts=1, msgid=<1012226395.140258.1402832586548.JavaMail.ABCPSCENTBB7$@ABCPSCENTBB7>, proto=ESMTP, daemon=MTA, relay=ABCPSCENTBB7.abc.ac.nz [123.45.60.199]

Jun 15 23:43:08 taag sendmail[4114]: s5FBh6qF004104: to=wert@exchange.abc.ac.nz,testab@exchange.abc.ac.nz,zenzenen@exchange.abc.ac.nz, delay=00:00:02, xdelay=00:00:00, mailer=esmtp, pri=120916, relay=exchange.abc.ac.nz. [123.45.60.104], dsn=2.0.0, stat=Sent (<1012226395.140258.1402832586548.JavaMail.ABCPSCENTBB7$@nb

Jun 15 23:43:08 taag sendmail[4114]: s5FBh6qF004104: to=croncroncroncron@imap.abc.ac.nz, delay=00:00:02, xdelay=00:00:00, mailer=smtp, pri=120916, relay=hauturu.abc.ac.nz. [123.45.1.17], dsn=2.0.0, stat=Sent (s5FBh8g3004823 Message accepted for delivery)

Jun 15 23:43:08 taag sendmail[4114]: s5FBh6qF004104: to=croncroncron@abcabcpu2.abc.ac.nz, delay=00:00:02, xdelay=00:00:00, mailer=smtp, pri=120916, relay=abcabcpu2.abc.ac.nz. [123.45.1.14], dsn=2.0.0, stat=Sent (s5FBh8Cn012207 Message accepted for delivery)

Jun 16 10:23:50 taag sendmail[7686]: s5FMNo62007686: from=<testab@abc.ac.nz>, size=4004, class=0, nrcpts=3, msgid=<46BBFB020B9AF54FA0C2DF26B796975A8684E1F9@Lewis.autuni.abc.ac.nz>, proto=ESMTP, daemon=MTA, relay=ali.abc.ac.nz [123.45.60.130]

Jun 16 10:23:53 taag sendmail[7701]: s5FMNo62007686: to=dl@abcabcpu2, delay=00:00:03, xdelay=00:00:00, mailer=smtp, pri=184004, relay=abcabcpu2.abc.ac.nz. [123.45.1.14], dsn=2.0.0, stat=Sent (s5FMNrsR019233 Message accepted for delivery)

Jun 16 10:23:53 taag sendmail[7701]: s5FMNo62007686: to=aaaa@x-files.abc.ac.nz,abcabcab@x-files.abc.ac.nz, delay=00:00:03, xdelay=00:00:00, mailer=smtp, pri=184004, relay=x-files.abc.ac.nz. [123.45.1.5], dsn=2.0.0, stat=Sent (s5FMNrjO018927 Message accepted for delivery)

Jun 15 23:43:08 taag sendmail[4114]: s5FBh6qF004104: to=croncroncron@abcabcpu2.abc.ac.nz, delay=00:00:02, xdelay=00:00:00, mailer=smtp, pri=120916, relay=abcabcpu2.abc.ac.nz. [123.45.1.14], dsn=2.0.0, stat=Sent (s5FBh8Cn012207 Message accepted for delivery)

Jun 16 10:23:50 taag sendmail[7686]: s5FMNo62007686: from=<testab@abc.ac.nz>, size=4004, class=0, nrcpts=3, msgid=<46BBFB020B9AF54FA0C2DF26B796975A8684E1F9@Lewis.autuni.abc.ac.nz>, proto=ESMTP, daemon=MTA, relay=ali.abc.ac.nz [123.45.60.130]

Jun 16 10:23:53 taag sendmail[7701]: s5FMNo62007686: to=dl@abcabcpu2, delay=00:00:03, xdelay=00:00:00, mailer=smtp, pri=184004, relay=abcabcpu2.abc.ac.nz. [123.45.1.14], dsn=2.0.0, stat=Sent (s5FMNrsR019233 Message accepted for delivery)

SendMail – Bad

Jun 15 22:01:35 taag sendmail[16757]: s5FA1VM4016721: SYSERR(root): header syntax error, line "?subject=Unsub+P814JUN15&body=Unsub+abchana@abc.ac.nz+from+the+P8>: "

Jun 16 09:13:57 taag sendmail[13209]: STARTTLS: write error=syscall error (-1), errno=32, get_error=error:00000000:lib(0):func(0):reason(0), retry=99, ssl_err=5

Jun 16 09:13:57 taag sendmail[13209]: STARTTLS: write error=generic SSL error (-1), errno=0, get_error=error:1409F07F:SSL routines:SSL3_WRITE_PENDING:bad write retry, retry=99, ssl_err=1

Jun 16 15:08:57 taag sendmail[30650]: s5G38Hwn030124: abcuabcu.abc.ac.nz.: SMTP DATA-1 protocol error: 250 OK

Jun 16 15:08:57 taag sendmail[30650]: s5G38Hwn030124: s5G38r27030650: DSN: Remote protocol error

Jun 16 21:46:38 taag sendmail[24294]: STARTTLS: write error=syscall error (-1), errno=32, get_error=error:00000000:lib(0):func(0):reason(0), retry=99, ssl_err=5

Jun 16 22:42:27 taag sendmail[9239]: s5GAg7e5009149: SYSERR(root): header syntax error, line "?subject=Unsub+P814JUN16&body=Unsub+abcwitt@abc.ac.nz+from+the+P8>: "

Jun 16 23:30:23 taag sendmail[21867]: s5GBUGTK021812: SYSERR(root): header syntax error, line "?subject=Unsub+P814JUN16&body=Unsub+abchana@abc.ac.nz+from+the+P8>: "

Jun 16 15:08:57 taag sendmail[30650]: s5G38Hwn030124: s5G38r27030650: DSN: Remote protocol error

Jun 16 21:46:38 taag sendmail[24294]: STARTTLS: write error=syscall error (-1), errno=32, get_error=error:00000000:lib(0):func(0):reason(0), retry=99, ssl_err=5

Appendix B – Earth Mover Distance Perl Library

```
package BMG::EarthMover;

#use strict;
#use warnings;

# Usage em/emd ( \@ka, \@kb ); pass it two arrays of the same length

use Exporter qw(import);
our @EXPORT = qw(em emd);

use List::Util qw[min max sum];

sub em {

#adapted from ftp://ftp.am-
utils.org/pub/osprof/analysis/stat_analysis.pl under GPL

    my @bins1, @bins2;
    my $tot1, $tot2, $j, $carrying, $emd, $max;

    @bins1 = @{ $_[0] };
    @bins2 = @{ $_[1] };

    $tot1 = $tot2 = 0;

    $tot1 = sum(@bins1);
    $tot2 = sum(@bins2);

    $carrying = $emd = 0;
    for ( $j = 0 ; $j <= $#bins1 ; $j++ ) {
        my $a = $bins1[$j] / $tot1;
        my $b = $bins2[$j] / $tot2;
        $carrying += $a - $b;
        $emd += abs($carrying);
    }

    $max = @bins1;
    return ( $emd / $max );
}

sub emd {

#adapted from http://commons.apache.org/proper/commons-
math/jacoco/org.apache.commons.math3.ml.distance/EarthMoversDistance.j
ava.html under Apache License, Version 2.0

    my @bins1, @bins2;
    my $tot1, $tot2, $j, $carrying, $emd, $max;

    @a = @{ $_[0] };
    @b = @{ $_[1] };

    my $lastDistance  = 0;
    my $totalDistance = 0;
    my $lena          = @a;
    my $lenb          = @b;

    for ( my $i = 0 ; $i < $lena ; $i++ ) {
        my $currentDistance = ( $a[$i] + $lastDistance ) - $b[$i];
        $totalDistance += abs($currentDistance);
```

```
        $lastDistance = $currentDistance;
    }
    return $totalDistance;
}

1;
```