UPPSALA
UNIVERSITET

IT 13 080
Examensarbete 30 hp
November 2013

Automatic Log Analysis using Machine Learning

Awesome Automatic Log Analysis version 2.0

Weixi Li

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Automatic Log Analysis using Machine Learning

Weixi Li

Many problems exist in the testing of a large scale system. The automated testing results are not reliable enough and manual log analysis is indispensable when automated testing cannot figure out the problems. However, it requires much expert knowledge, costs too much and is time consuming to do manual log analysis for a large scale system. In this project, we propose to apply machine learning techniques to do automated log analysis as they are effective and efficient to big data problems. Features from the contents of the logs are extracted and clustering algorithms are leveraged to detect abnormal logs. This research investigates multiple kinds of features in natural language processing and information retrieval. Several variants of basic clustering and artificial neural network algorithms are developed. Data preprocessing is experimented before feature extraction as well. In order to select a suitable model for our problem, cross validation and F-score are used to evaluate different learning models compared to automated test system verdicts. Finally, the influences of factors that may affect the prediction results such as single or mixed test cases, single or mixed track types and single or mixed configuration types are verified based on different learning models.

Handledare: Johan Karlsson
Ämnesgranskare: Kjell Orsborn
Examinator: Ivan Christoff
IT 13 080
Tryckt av: Reprocentralen ITC

Acknowledgements

I would like to show my greatest gratitude to my supervisor Johan Karlsson, manager Jens Wictorinus and reviewer Kjell Orsborn for always giving me nice suggestions and great ideas during these days. And thanks very much to my parents, my boyfriend and all friends and colleagues that helped me.

Contents

List of Tables

List of Figures

1. Introduction

With the growing scale and complexity of the IT systems, debugging and testing become more difficult. As it is very difficult for the verification engineer to have an insight in every piece of work that the developers do, the test results based on the automated test scripts are not completely reliable. The logs generated by the systems, in another point of view, reveal the states of a running system and contain a wealth of information produced by the system to support diagnosis, hence, the log analysis is a vital method of detecting failures or problems in a large system. It is, however, not realistic and practical to analyze huge amounts of logs manually. In a Big Data world, using machine learning techniques to do automated log analysis becomes interesting.

This project aims to extract effective features from the free text contents of RBS (*Radio Base Station*) [36] functional test loop logs in Ericsson, detect the abnormal logs automatically through machine learning techniques and investigate the effectiveness of different solutions.

In the following chapters, related work in the relevant researches will be introduced, and an overview of our learning system AALA 2.0 will be given followed by the detailed introduction and discussion of different features and algorithms. Experiment results of different case studies will be presented in Chapter 6. The final chapter will conclude the major findings and improvements of this project, and discuss the limitations in this project as well as the future work.

1.1 Background and Motivation

Testing results can not be trusted all the time.

In a complex system, like RBS in Ericsson, software and hardware are developed by multiple developers and updated and changed frequently. Generally, the verification engineers are not as familiar with all the system details as the developers. Furthermore, no one can foresee all problems. Many unknown problems will occur with a growing and continuous updated system. It is hard for the verification engineers to create perfect automated test scripts that can detect all potential problems. For example, in the RBS test loops in Ericsson, the test cases passed in 2hr loops do not pick up all the errors in the software and the test cases in later loops might fail. Automated testing is likely to ignore some important errors or faults, but perhaps, these abnormal behaviours can be discovered in the logs.

Manual log analysis is hard for large systems.

In reality, analyzing the logs manually plays an indispensable role when automated testing cannot figure out the problems. However, a complex system usually involves a wide variety of techniques which need multiple experts of different fields to cooperate. When people decipher logs manually, the lack of operational context makes many log messages ambiguous. And a single failure type may produce varying alert signatures in the log. Furthermore, the meaning or characters of the logs can be very abstruse with abbreviation and hexadecimal digits and can drastically be altered even though only minor configuration changes [26].

With the CI (*Continuous Integration*) [35] process introduced, a large-scale system usually dumps tons of logs everyday and the majority of them are not interesting for the testers. It costs too much and is time consuming if we train a lot of experts to identify all interesting logs correctly and look into each of them manually. Therefore, that is done by testers and usually, testers select some suspicious log files instead of looking into all logs and they use simple tools or commands (e.g. grep) to search some predefined keywords (e.g. error, failed, abort) which may help identify problems based on personal experiences.

In order to detect problems through analyzing logs manually, the verification engineer must understand the behaviours of the system and the logs it produces. But the number of existing software and growing

new software in the system is quite large and the logs of different software are huge different. It is hard to find such an expert having all necessary knowledge. Therefore, the inefficiency and difficulties of manual log analysis make automated log analysis desirable.

Using explicit programming is hard.

A large-scale system is usually composed of various parts, and those parts may be developed with different programming languages. Despite of the noble efforts of the BSD syslog standard [22] and other standards, the contents of different components' logs can be greatly different. There is no output structure standard for all log files, and it is, therefore, hard to implement one single log analyzer for the development of a large-scale system (e.g. RBS) through explicit programming. Even if we can implement specific log analyzer for a specific software, it cannot be commonly used in a system under development without frequent updating. Additionally, as mentioned in the previous paragraph, even as for human beings, it is hard to detect problems from tons of logs correctly, and thus it is nearly impossible to implement a perfect log analyzer through explicit programming by human beings.

Machine learning might help.

Training many people to become such an expert mentioned previously costs a lot and is inefficient, but people may consider to train a computer system to learn by itself, which is exactly the concerning topics of the machine learning techniques. Machine learning techniques can automatically handle large scale data processing and generalize experiences from learning patterns on a set of training data and apply on a new data set slightly different from the training data without retraining. Many machine learning techniques have been successfully leveraged in some complex problems, ranging from learning to detect fraudulent credit card transactions, learning users' reading preferences to learning of autonomous vehicles to drive on public highways [24]. Machine learning algorithms can automate the process to do log analysis and analyze all logs much faster than humans.

Additionally, machine learning does not rely on explicit programming. It does not need to know all the truth about the problems. In machine learning, unsupervised learning [24] can learn from the structures of the training data without given true answers. Therefore, it is possible to find hidden problems in the logs of different formats and contents by machine learning. Also, it is not a necessary condition that the developer of a machine learning system has any expert knowledge in log analysis but is a plus in finding effective features.

1.2 Major Work and Contributions

Ericsson has been committed to pursue automated log analysis using machine learning techniques. This thesis is a continuation of an existing prototype-AALA 1.0 (*Awesome Automated Log Analysis, version 1.0*) [11]. The details of AALA 1.0 will be introduced in Chapter 2. A proof of concept study has been done, but there are many shortcomings due to time limitations. This project uses some parts of AALA 1.0 but is not limited by its ideas.

The major work of AALA 2.0 is listed as below.

1. We investigate new features and machine learning algorithms other than those in AALA 1.0.
   a) We extract TF-IDF [14] and word bigram as new features from the unstructured logs and investigate different combinations of them and the features in AALA 1.0 instead of just putting all kinds of features together in the feature matrix.
   b) We apply the combinations of self-organizing feature maps (*SOFM*) [16], DBSCAN [9] and K-means [21] to detect abnormal logs and develop more effective variants, e.g. SOFM-DIST, SOFM-COUNT, SOFM-DBSCAN and Multi-layer SOFM.
2. We investigate different combined solutions on different data sets.
   a) We investigate some ways to do data preprocessing before feature extraction.
   b) We investigate the proper way to use principal component analysis (*PCA*) [27] to do dimensionality reduction for our features.

c) We experiment with more than 100 different solutions (including with and without data preprocessing, different ways to do PCA, different combinations of features and different variants of algorithms) on different data sets (including different single test case and mixed test cases, single track type and mixed track types, different single configuration type and mixed configuration types).

3. We build a more systematic learning process. We do cross validation to select a model with better generalized ability and use a suitable metric *F-score* [33] to evaluate our models.

Our contributions are as follows.

1) We prove that it is possible to detect abnormal logs based on the features extracted from the contents and unsupervised learning algorithms.
2) We describe some novel variant learning methods for automated detection of abnormal logs, which perform better than the original basic methods.
3) Our recommended solution for general cases can detect almost all failed logs in the automated testing results and find unknown problems as well.
4) AALA 2.0 improved AALA 1.0 and got a much better result.
5) We investigate the results of logs of different types and find that different test cases could be a factor that affects the detection results.

1.3 Thesis Structure

The rest of this thesis report is structured as follows, and due to rules in Ericsson, the report omits all confidential information in the following chapters. Besides, some references to the documents in Ericsson are not public to the outsiders.

In Chapter 2, the previous work on the feature extraction and pattern mining of logs, and the state-of-art of detecting anomalies and failure diagnosis from logs by using machine learning techniques are reviewed in general.

In Chapter 3, an overview of our learning framework is described firstly and the workflow of our approaches is presented.

In Chapter 4 , the investigation of different features extracted from the logs in this project is introduced.

In Chapter 5 , different machine learning algorithms and their variants investigated in this project are introduced.

In Chapter 6, some case studies are introduced and the results are presented. Multiple factors that may affect the learning results are mainly compared and the testing results of the most proper algorithm are shown.

In Chapter 7, a summary of the thesis work and the improvements in this project compared to AALA 1.0 are concluded with as well as the discussion of the major limitations and possible future work in this project.

2. Related Work

In this chapter, the state-of-art of this research topic is described. In section 2.1, some researches conducted so far on extracting features and mining the patterns of logs which could be used in the machine learning techniques are reviewed. In section 2.2, the existing machine learning techniques used in this topic are explored. Section 2.3 mainly introduces the previous work in AALA 1.0.

2.1 Feature Extraction from Logs

Logs have a lot of different types, e.g. sever logs, database logs, event logs etc. The contents of logs include numerical data and non-numerical data as well. For some kind of logs, like server logs, those numerical data (e.g. CPU load, memory) is sufficiently representative as features. But the meaningful non-numerical data (e.g. the indicators about the state of systems) are also interesting. The features reflecting the text contents of the logs are the focus of our research.

In the log analysis area, there is no generally accepted standard in the feature selection. Although it is still under investigation, many different methods have been tried and some representative features from logs have been studied.

As the logs can be viewed as textual data, some natural language processing techniques can be applied. Many studies extract N-gram frequencies [4] from the logs. An n-gram is a contiguous sequence of n items from a given sequence of text or speech [15]. It is a common feature when doing text categorization. For example, [4] did language classification and subject classification of newsgroup articles based on calculating and comparing profiles of N-gram frequencies. There is also one related study in abnormal detection of logs. [30] used N-gram frequencies of network logs from Apache servers as features to do intrusion detection. It is usually to take a word rather than a character as a gram in the text mining area, and use contiguous sequences of n items instead of any permutation of all items.

TF-IDF [14] is another famous feature in natural language processing area. Not only is the frequency of one word in a specific document considered, but also the discriminative power of one word in a document collection is included. [37] applied this measure to its message count features, and improved the detection accuracy of abnormal logs.

Additionally, many researches use data mining techniques to mining patterns from the data as features. [1] created a dictionary of event types by a sequential text clustering algorithm and identified individual sets of messages in a log file that belong to one process or failure through the PARIS algorithm (*Principle Atom Recognition In Sets*). In this paper, the algorithms are aimed to find relevant information among the vast amount of messages in the logs through creating the dictionary that represents the event types from the messages.

Parts of the log analysis researches extract structure information from logs by explicit programming. They define specific important attributes for their logs and extract them as features. The study [37] pointed out that although the logs are non-structured data but the source codes generating these logs have static templates. Two kinds of attributes are representative, that is, the state variables which should have a small constant number of distinct values and the object identifiers which should identify program objects and have multiple values. In [37], the state ratio vectors and message count vectors were treated as features.

2.2 Machine Learning Techniques

Many machine learning techniques could be used in the automated detection of abnormal logs. Some of them have been investigated in this area while the others have been studied in some related fields, such as

text mining, anomaly detection. This section concentrates on introducing the state-of-art of the applications of machine learning on relevant fields. These techniques are categorized into classification techniques, clustering techniques and statistical techniques.

Classification Techniques

In machine learning, classification [13] is a process to approximate a function mapping a vector into classes by looking at input-output examples of the function. It is a kind of supervised learning techniques which means it needs labeled data to supervise the learning process. [38] used Support Vector Machines (*SVM*) [34] to classify system call sequences of solved problems, but the correlations between system behaviors and known problems cannot be applied to detect unknown problems. Bayesian networks [12] is another commonly used classification technique in anomaly detection, for example, the paper [2] proposed a kind of variant Bayesian network to do network intrusion detection.

The advantages of supervised learning techniques are efficient and fast, as they get the 'correct' answers during training, that is, the clear feedbacks help them learn quickly. Besides, it is more powerful to detect the known problem since the patterns have been taught. But the disadvantages are obvious that it is hard to get enough reliable labeled data. First, it is hard to provide absolutely normal data (i.e. containing no problems), and if some erroneous data are labeled as normal data, it increases the false positive error. Secondly, as even the human expert cannot detect every kind of problem, supervised learning will fail to detect a totally unknown problem. Finally, the supervised learning system can never be superior to human experts, since the goal is to imitate the experts [8].

In this project, it is nearly impossible to get enough labeled data to do supervised learning and the stakeholders are interested in those unknown problems as well. Therefore, this project did not use classification techniques.

Clustering Techniques

Since the complete clean data (i.e. containing only normal information) are almost impossible to be guaranteed, many researches use the unsupervised learning methods. Majority of them adopt the clustering techniques. Clustering [13] is the process of grouping the data into classes or clusters, so that objects within a cluster have high similarities in comparison to one another but are very dissimilar to objects in other clusters.

The researches usually have two different kinds of assumptions to apply clustering techniques .

The first kind is assuming that the anomalous data can form clusters by themselves [5]. K-means [21] is a classical partitioning method which partitions a set of n objects into k clusters, so that the resulting intracluster similarity[1] is high, but the intercluster similarity[2] is low. The major disadvantage of K-means is that the input parameter k (i.e. the number of clusters) needs to be set by people. A suitable k is crucial for the final results but finding a optimal k is an NP-hard problem [3]. [10] proposed an improved algorithm called Y-means to overcome the parameter selection problem. Y-means automatically selected the value of k through splitting and merging clusters, and set a threshold to separate normal clusters and abnormal clusters. It solves the shortcomings of K-means but still has the selection problem of parameter threshold t. As for the logs, the categories of logs are complicated and difficult to decide. Only two categories of either normal logs or abnormal logs are desired for this project. How to map the clusters of K-means into two categories is a problem that needs to be solved in our project.

Self-Organizing Feature Maps (*SOFM*) [16] is another popular clustering method. SOFM is also called Kohonen map or network, which was first described as an artificial neural network by the Finnish professor Teuvo Kohonen [16]. SOFM can discover the significant patterns in the data by capturing the topology of the original input data. When SOFM is used in the anomaly detection area, the researchers usually assume that the normal data instances are quite close to their closest cluster centroids, but the abnormal instances are far away from their closest cluster centroids [5]. [7] used SOFM for the fault monitoring in signal systems. SOFM has also been leveraged in the intrusion detection. [18] and [29] aimed to classify regular and irregular intrusive network traffic for a given host via SOFM. The abnormal high value of the distance

[1] Intracluster similarity means the similarity of those points in the same cluster.
[2] Intercluster similarity means the similarity between different clusters.

of the winning neuron with respect to the input vector was treated as an irregular behaviour. [25] is a more relevant research on log analysis. That is a case study about using SOFM on the server log data. But their training data are only the numerical data (e.g. CPU load, User-time) rather than the text contents in our case. The advantage of SOFM is that it is independent of the data distribution and cluster structure, but the shortcoming is that if the data do not have constant variance, the global threshold will be too low for the data of larger variation and leave these anomalies undetected [8]. [17] proposed a technique to use local thresholds for each cluster. Also, SOFM is not efficient enough for large data processing, especially on-line processing. [40] proposed a more efficient algorithm combined with the fast nearest-neighbor searching strategy. In this thesis project, the off-line learning is the only candidate.

The second kind of assumption is that the anomalies do not belong to any cluster after training [5]. The clustering techniques that do not force every data instance to belong to one cluster have been widely used under this assumption for anomaly detection. For example, DBSCAN [9] is a kind of density-based clustering technique based on connected regions with sufficiently high density. As it is based on the density of data, DBSCAN can discovery clusters with arbitrary shape and is anti-noise, so that it can find the clusters that K-means cannot find [33]. DBSCAN has the disadvantage as SOFM, which is not capable to handle local density variation within the cluster. [28] proposed a density varied DBSCAN algorithm which overcomes this shortcoming. And because the definition of density on high-dimensional data is meaningless, DBSCAN is usually not suitable for high-dimensional data.

Clustering is a suitable type of machine learning algorithm for our problem since not enough complete clean labeled data can be guaranteed in this thesis work.

Statistical Techniques

Since the log analysis could be treated as an anomaly detection problem, many anomaly detection techniques have been explored. Generally, the statistical anomaly detection techniques leverage a statistical model to formulate the problem, and the anomalies are assumed to be in the low probability regions of the model.

Principal component analysis (*PCA*) [27] has been studied in many fault detection problems. PCA separates the high-dimensional data space into two orthogonal subspaces. The significant changes in the residual portions of a sample vector indicate that it is an anomaly. [6] used PCA for processing monitoring; [19] used PCA for diagnosing network-wide traffic anomalies. In the log analysis problem, [37] used PCA to detect anomalous logs. PCA is a kind of effective method when dealing with high-dimensional data. But the shortcoming of this method is that it needs a threshold to separate the normal data and abnormal data. To select a suitable threshold is hard and needs prior knowledges.

Statistical techniques could be a candidate solution for our problem.

2.3 Brief Introduction to AALA 1.0 (Awesome Automatic Log Analysis version 1.0)

In this section, I will introduce the AALA 1.0 [11] briefly. In AALA 1.0, the author mainly extracted character N-gram, timestamp statistics, word count, simplified message counts and message templates from the data set of two types of logs. After feature extraction, the author did normalization and dimensionality reduction with PCA to those features. Finally, the author built a simple framework to learn a model based on basic K-means, DBSCAN clustering or PCA-based anomaly detection with those normalized and reduced features.

AALA 1.0 used a feature matrix combined of character N-gram, timestamp statistics and word count to learn models by basic K-means and DBSCAN. In K-means, the author used the cost function plot to select a suitable k and found one cluster with a single log that was obviously abnormal but recorded as passed by the test system. But choosing the k via plot is not practical for the automated solution in reality and it is necessary to analyze each cluster to see if this cluster is normal or abnormal, which is not automated either. Besides, the results tend to cluster the logs into different types and there is no obvious distinguish between normal data and abnormal data. DBSCAN in AALA 1.0 tried to group the anomalies into one

or more clusters after excluding the outliers rather than treated the anomalies as outliers, and because of limited time, there was no suitable parameters and the preliminary result was not good.

Since the results of simple features and basic clustering algorithms were not satisfying enough, AALA 1.0 tried to discover the structures of log files via explicit programming. The author tried to extract similar features as [37] from logs directly, since it is hard to get source codes and do static analysis. But because of the shortage of time, this solution was not completely investigated. As the structures were extracted only from the logs, those features were limited and less accurate than those in [37]. AALA 1.0 applied the features directly to the PCA-based anomaly detection algorithm without dimensionality reduction, but it did not yield any interesting result and the author claimed that it is better to pursue implementing the full features of [37].

A dataset of logs from two test cases was investigated in AALA 1.0 and there was no cross validation and suitable evaluations due to limited time. AALA 2.0 remedied these flaws and tested on multiple different data sets with more investigations on features and algorithms.

3. Overview of AALA 2.0

In this chapter, an overview of the whole process of our learning system is given. The details of the problems in this project are analyzed in the first section and then the general process to solve this kind of problem is introduced. At last, the research methodology used in this project is presented.

3.1 Class Imbalance Problem

The aim of our project is to detect abnormal logs from the logs produced by RBS testing loops. From the section 1.1, we know that these logs from passed test cases cannot be trusted completely, but we can still get some useful information from the history data. For example, the failed logs own low percentage of the total amount of logs.

Generally, the problems Machine Learning solved are classification and clustering. Since our training samples have no reliable labels which show the categories they belong to, this problem is viewed as a classification problem learning by clustering methods, which means that it is necessary to use clustering methods to cluster our logs and classify these clusters into two categories, i.e. normal logs and abnormal logs. In another perspective, our problem is a kind of abnormal detection problem due to the low percentage of abnormal logs, and the logs containing system's abnormal behaviours are treated as anomalies. Abnormal detection is usually a class imbalance problem as well since the anomalies are much few than the normal ones. Generally, a class imbalance problem has the following characteristics [33]:

Problem 1 If there are multiple attributes, it is necessary to figure out a definition of the abnormal item via the values of multiple attributes.

Problem 2 An item is abnormal compared to all items, but it is normal compared to its neighbors.

Problem 3 There is a disproportionate number of instances that belong to different classes. Accuracy and error rate may not be well suited for evaluating models derived from imbalanced data sets.

With regard to our log analysis problem, the following problems are required to be solved.

Firstly, which combination of features is effective to represent the abnormal behaviours should be figured out, as a log file may be normal in some features but abnormal in other features, and it is possible that all the values of its features are normal if they are treated separately, but the log file is abnormal if these features are treated together. Therefore, different combinations of features should be tried in the experiments in order to find the suitable definition of the abnormality of logs.

Secondly, those factors that may affect the perspective of abnormality should be found out. Because a log file may be very different compared to the majority of logs, in this situation, we cannot confirm this log file is abnormal or not. It is possible that this log file is from a special test case in which all logs are huge different from the logs in other test cases and the number of logs in this test case is few than others, so it looks abnormal compared to those logs in other test cases. In fact, the log file is normal compared to the other logs in its test case. Therefore, it is necessary to verify that if some factors, like logs from different test cases, may affect the clustering results in our project.

Thirdly, in order to select a suitable model, a suitable metric is required to do evaluation. Because in this project, it is not possible to find real experts to do manual log analysis, the results can only be compared with the test system verdicts. And since the number of failed logs is much less than the number of passed logs, accuracy and error rate are not good metrics for our evaluation. For example, if there are 100 logs in total and only one log file is abnormal, we guess all logs are normal so that we can get an accuracy of 99%. In fact, this guess is not a good model as we do not detect any abnormal log. Therefore, it is important to figure out suitable metrics rather than accuracy and error rate.

3.2 Workflow of AALA 2.0

In this section, an overview of the learning framework is shown in Figure 3.1. AALA 2.0 complies with the general method of building a learning model, that is, it has feature extraction phase, followed by model learning phase and generalization ability tests by new test set. Also, it tunes some parts as well according to the requirements of our problem.

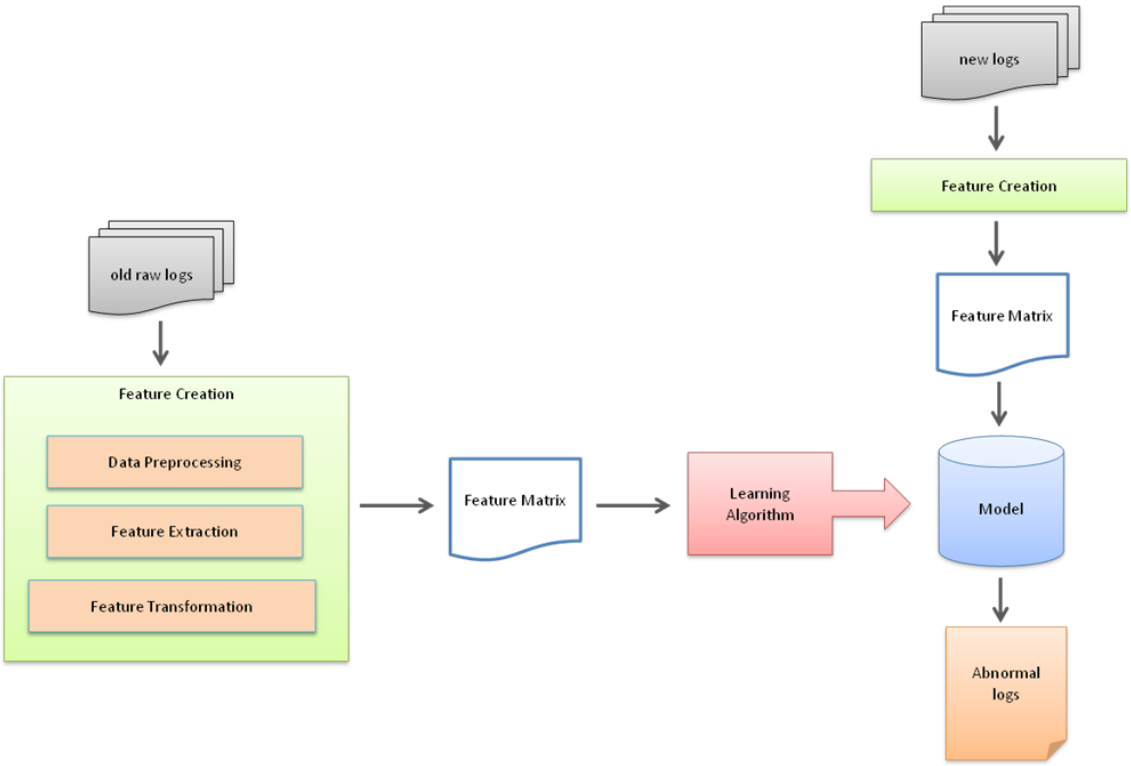


Figure 3.1: Overview of the learning system AALA 2.0

3.2.1 Text Normalization

As mentioned before, the format of our logs is not fixed. But they have some common characteristics, for example, each message in one log is started by a time stamp with a fixed format and other characteristics like the logs are written in English and composed by upper and lower case letters, digits and special characters.

Inspired by Natural Language Processing (NLP) [31], it might be helpful to do some text normalization to the raw logs before extracting features. This step can be omitted in some approaches. The following list shows our text normalization jobs.

Replace timestamps Use a special symbol to replace the whole timestamp before each message.
Replace digits Use a special symbol to replace any digit in the log file.
Lower cases Change all upper case letters into lower case.
Remove special characters Remove all special characters, including punctuations, and only keep letters and digits.

Timestamp information is important according to the expert knowledge of the verification engineers, because the automated testing executing too quick or too slow usually indicates the occurrence of exceptions. Therefore, they will be extracted as a separate kind of features. Regarding some other kinds of features themselves, e.g. TF-IDF, timestamp information is meaningless, therefore, they are replaced with placeholders. As for the other three kinds of normalization, their effectiveness for our goals will be verified.

3.2.2 Feature Extraction

After text normalization, the system can extract effective features via the feature extraction module. It could be one or multiple kinds of features. One kind of feature can include multiple attributes. In most cases, one kind of extracted feature will be combined into a matrix. In this matrix, each row stands for one log file, and each column stands for one attribute of this kind of feature.

When these features are input into the modules in Matlab, firstly, they should be combined into a big feature matrix if there exists more than one kind of extracted feature. Similarly, in this feature matrix, each row stands for one log file, and each column stands for one attribute of these kinds of features.

3.2.3 Feature Transformation

Data Normalization

Doing data normalization to the feature matrix is to make the whole data set have consistent statistical properties. Different kinds of our features may have different range of values. In order to avoid those features with larger range of values being dominant, data normalization is usually necessary. For example, the range of time duration of one log file is small, sometimes only takes several seconds, while the range of one word's frequency may raise up to several hundreds. If we compare two logs with these two kinds of features, the words' frequency features can sway the outcome.

In this step, the common data normalization technique to normalize the whole data set with a same range of values is used.

Dimensionality Reduction

Dimensionality reduction is a common technique used to reduce the number of variables in Machine Learning. In our project, the amount of extracted feature attributes is large. For example, if we take the occurrence frequencies of each word in the log files as an example, the amount of words is very large. Also, different logs may differ a lot, that is, some words may only occur in certain types of log files, and as for the majority of log files, the values of these words are zero. It easily leads to 'dimensionality disaster' [33]. The high-dimensional data is usually located in a sparse space. As for the clustering algorithms, the definition of density and distance between two data points becomes meaningless, which is harmful to the generalization of models and the outcomes of clustering become worse. Dimensionality reduction is also helpful to remove the irrelevant feature attributes and reduce the noises. In the meantime, it is easier to learn a model with lower dimensionality with clustering techniques. Finally, dimensionality reduction can increase efficiency and decrease the consuming of time and memory resources. In order to verify the effectiveness of dimensionality reduction in our problem, both solutions with and without dimensionality reduction are tested in the experiments and from the results, we can see that it is more effective and efficient with dimensionality reduction.

3.2.4 Learning a Model

After the feature transformation, the system puts the feature matrix into one of the learning methods, and it will output a learnt model. If this model is the most suitable for our problem, then this model can be directly applied to new log files without retraining and the prediction results will find out those abnormal logs in these new log files. We do not retrain every time when there are new log files that are slightly different than those training logs. It is only need to retrain a model when there are new test cases or the old test cases are changed totally. We learnt more than 100 models of different solutions.

3.2.5 Evaluation and Selecting the Model

Among those learnt models, a suitable one should be selected as our final solution. Stratified k-fold cross validation [33] is used in this project to select proper learning parameters and select the best model. A

common setting $k = 10$ according to [23] is chosen, and then the data set is divided into two sets, training set and test set. The training set is used to learn models, and each model is applied on the test set to compute the F-scores [33]. Section 6.1 will introduce our metrics in detail. Finally, the model with the highest average F-scores in the 10 test sets is chosen. The average F-score could be an estimation of the generalization ability of one approach and can be used to compare different approaches.

F-score is used as a guideline to evaluate different models and different parameters, and in the experiments it is calculated by comparing the prediction results with test system verdicts due to shortage of experts. That is, test system verdicts are not used during learning. Also note that a high F-score does not necessarily indicate a best model in our case.

3.3 Methodology

In section 3.1, we have already known our problems. In this section, the methodology to solve these problems is explained.

Firstly, through studying the related work in Chapter 2, the feature candidates and learning algorithm candidates are chosen. As far as the Problem 1 mentioned in section 3.1, it is necessary to try 22 different combinations of features listed in section 4.2.2 on the same data set. And because some learning methods may be good at learning certain features and some may be good at others, 10 combinations of different algorithms on each kind of combination of features are needed to try.

Secondly, due to limited size and unknown verdicts in test system of old training data set in AALA 1.0, new data sets are decided to be used in AALA 2.0. And by analyzing the results in AALA 1.0, some possible factors that may raise Problem 2 are found. In the test loop, there are a lot of test cases and each test case's object is different. Also, the test loop includes different tracks for different testing purposes and the testing objects contain different configuration types. Therefore, the experiments in this project will test the system on the data sets with different single and mixed test cases, track types or configuration types. Those factors may confuse the clustering results and dominate the categorization. For example, some test cases may output huge different logs compared to other test cases and then the definition of abnormality will be not correct when learning a model. Therefore, experiment results are used to prove their effects. And different sizes of each data set are tested to see how many logs are enough for training a model with stable results as well.

Additionally, not all the steps in the workflow described in section 3.2 are necessary for learning a good model. Therefore, both learning process with and without some parts of the workflow are tried in the experiments. When the raw log files are transformed into real training matrix, the effects of parts of those preprocessing jobs are required to be checked through experiments. This thesis work runs experiments both with and without text normalization and dimensionality reduction and tries different ways to do dimensionality reduction. Although these preprocessing jobs help a lot in general machine learning problems, their effectiveness in this project is needed to be confirmed.

Finally, some literatures related to the Problem 3 are studied and a suitable metric F-score is selected to evaluate different models, which will be introduced in section 6.1.

The general results will be discussed in Chapter 6 by showing some case studies.

4. Feature Investigation

Features play an important role in Machine Learning, because only the features truly reflecting the data's nature can generate a good model. However, how to identify good features is a big problem for our project due to shortage of expert knowledge of log analysis. Therefore, only some candidate features focusing on time statistics and text information can be selected and then the proper feature combination is selected via experiments. Firstly, some basic features are introduced. In the second part, the key insights on the feature investigation in this project are introduced.

4.1 Basic Features

We investigated the following three kinds of basic log features, among which character bigram, timestamp statistics and word count have been applied in AALA 1.0 as well. We rewrote these features into our Python Preprocessor and implemented some new features.

4.1.1 Timestamp Statistics

As have been pointed out in Section 3.2.1, timestamp information is a common element in normal logs. Experienced verification engineer found that the total executing time of the testing is a significant criteria to judge the abnormality of logs. From the experience of testers, it is usually abnormal that one of our test cases takes greater than 5-10 minutes and if it is over 30 minutes, it is clearly a fault. Therefore, the timestamps in each log file are collected, and the time differences between any adjacent messages are computed. Besides the total executing time, the variance, the maximum, and the average value of these time differences are also interesting.

Let $t(l,m)$ be the timestamp of the message m of the log file l. Let $\Delta t(l,m)$ be the time difference between message m and its previous message $m-1$, that is,

$$\Delta t(l,m) = t(l,m) - t(l,m-1) \tag{4.1}$$

Let $|l|$ be the number of messages in log file l. $T_\Delta(l)$ designates the set of all time differences of log file l, that is, $T_\Delta(l) = \{\Delta t(l,1), \Delta t(l,2), \ldots, \Delta t(l,|l|)\}$. The timestamp statistics matrix T has four columns as below.

$$T(l,*) = [mean(T_\Delta(l)), var(T_\Delta(l)), sum(T_\Delta(l)), max(T_\Delta(l))] \tag{4.2}$$

4.1.2 N-gram

Another criteria one experienced verification engineer will use is to search any matched error patterns in the logs. These patterns are actually some key words, e.g. error, warning, crash, unexpected. But it is hard to say that the logs containing these key words are definitely abnormal, so these key words are not directly used to judge the logs in this project.

Some basic features in the field of natural language preprocessing are explored to reflect the nature of these key words. ***N*-gram** is one of them. An n-gram is a sequence of n items combined from a given string

or text [4]. **Items** have no specific assignment. It can be letters, words or even syllables if they are applied in speech. And an n-gram could be any combination of items. For example, a character bigram ($n = 2$, item refers to character) of the word "TEXT" could be "TE", "EX", "TX", "TT", etc. In this project, two kinds of n-gram, **character bigram** and **word bigram**, are used.

character bigram

In the **character bigram**, all printable characters are viewed as items, e.g. 1,a,A,#,$,[,). All possible combinations of two characters are used as the bigram dictionary. For example, "12", "21", "a(", ")a". Let $CB(l,g)$ designate the number of occurrence of character bigram item g in log file l.

word bigram

In the **word bigram**, firstly, it is necessary to generate a word bigram dictionary using the existing tokens segmented on spaces in the preprocessed logs. But only the combinations of two consecutive tokens are interesting for this project. For example, the word bigrams of the text " Awesome Automated Log Analysis " can only be " Awesome Automated ", " Automated Log " and " Log Analysis ". And then the number of occurrence of word bigrams in each log file is counted to build a matrix. $WB(l,g)$ designate the number of occurrence of word bigram item g in log file l.

As for time and resources limitation, only $N = 2$ situations are tried in this project, but the algorithms are easy to generalize to other situations if you want.

4.1.3 Word Count

Word count is another common feature in natural language preprocessing. It counts the number of occurrences of words from a dictionary in each log file. Our dictionary contains all tokens segmented on spaces from the preprocessed log files. Let $W(l,w)$ be the number of occurrences of word w in the log file l. This notion is also called as **term frequency** in other field, but in order to avoid confusion with TF in section 4.2.1, so it is called word count in this report.

4.2 Advanced Features

4.2.1 TF-IDF

TF-IDF [14] is an advanced feature based on the word count, which is commonly used in the field of information retrieval. **TF** (*term frequency matrix*) measures the association of a word with respect to a given document. Simply stated, it is a normalized word count. There are different ways to normalize word count [13]. In this project, the way of **relative word count** is chosen, that is, the word count versus the total number of occurrences of all the words in the document. Let $count(l,w)$ be the number of occurrences of word w in log file l and $|l|_w$ be the total number of occurrences of all the words in the log file l. We use the following formula to compute $TF(l,w)$:

$$TF(l,w) = \begin{cases} 0 & \text{if } |l|_w = 0 \\ \frac{count(l,w)}{|l|_w} & \text{if } |l|_w > 0 \end{cases} \tag{4.3}$$

IDF (*inverse document frequency*) represents the importance of one word. If a word appears in many log files, the importance of this word will be decreased due to its reduced discriminative power [13]. As these key words showing the abnormal behaviours usually appear in a small number of log files, IDF is an important feature to identify these key words.

There are also many ways to define IDF. In this project, the following way is used. Let $|L|$ be the total number of log files in the log file collection L and $|L_w|$ be the number of log files containing word w. $IDF(w)$ is defined by the following formula:

$$IDF(w) = \begin{cases} 0 & \text{if } |L_w| = 0 \\ \log \frac{|L|}{|L_w|} & \text{if } |L_w| > 0 \end{cases} \tag{4.4}$$

TF and IDF are combined to form the **TF-IDF measure** as below:

$$TF\text{-}IDF(l,w) = TF(l,w) \times IDF(w) \tag{4.5}$$

4.2.2 Combination

In the AALA 1.0, all the features are combined into a big feature matrix for learning. It is not a good way to investigate the features, because, in the beginning, it is hard to know if all the kinds of features are good to form an ideal model due to our limited expert knowledge and some of them may have negative effects on the result. At first, different combinations of these kinds of features are tried and the combination of features with the best result is selected.

The number of all possible combinations is $C_5^1 + C_5^2 + C_5^3 + C_5^4 + C_5^5 = 31$. Due to limited time, the following assumptions are made to remove certain of the combinations.

- Only using one kind of *n*-gram is enough.
- Timestamp statistics will not be considered alone.

Finally, there are 22 kinds of combinations as below to be investigated.

1. character bigram
2. word count
3. character bigram + timestamp statistics
4. character bigram + word count
5. timestamp statistics + word count
6. character bigram + timestamp statistics + word count
7. TF-IDF
8. character bigram + TF-IDF
9. timestamp statistics + TF-IDF
10. word count + TF-IDF
11. character bigram + timestamp statistics + TF-IDF
12. character bigram + word count + TF-IDF
13. timestamp statistics + word count + TF-IDF
14. character bigram + timestamp statistics + word count + TF-IDF
15. word bigram
16. word bigram + timestamp statistics
17. word bigram + word count
18. word bigram + TF-IDF
19. word bigram + timestamp statistics + word count
20. word bigram + timestamp statistics + TF-IDF
21. word bigram + word count + TF-IDF
22. word bigram + timestamp statistics + word count + TF-IDF

The most ideal combination of features will be presented in the Chapter 6.

5. Learning Algorithms

If we say that selecting proper features is the foundation to build a learning system, it is the learning algorithms selection that is the crucial phase to make good use of these features. As mentioned in Section 3.1, unsupervised learning and specifically clustering algorithms are used in this project. The basic clustering algorithms usually output more than two clusters, and it is necessary to map these clusters into two categories, normal and abnormal logs. In this chapter, the new variant versions of some basic clustering algorithms are introduced and some new combined algorithms based on them are presented. Due to the requirement of confidentiality, the implementation details are not provided here.

5.1 Basic Algorithms

5.1.1 K-means Variants

K-means [21] is commonly used for clustering problems. The similarity between two objects is measured via the distance between two points. K-means divides n data points into k clusters that makes the sum of distances between each data point to its centroid in each cluster minimize. The objective function is the SSE (*Sum of the Squared Error*) [39].

Let C_i be the set of all data points in the ith cluster and c_i be the centroid of the ith cluster. p designates any data point. The *SSE* is:

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} ||p - c_i||^2 \tag{5.1}$$

The original K-means algorithm computes the k centroids iteratively with k randomly chosen initial centroids.

Since there are different types of logs, it is easy to understand that the number of categories after clustering is greater than two. But it is hard to predict how many categories the log collection contains, that is, the number of clusters k is hard to define. The cases of k to be 2 are tested but the results are bad, see Figure 5.2. The red point refers to abnormal log and the blue one refers to normal log. Figure 5.1 is the visualization of the verdicts of the testing results on this sample data set. The red point refers to failed log and the blue one refers to pass log. If only $k = 2$ is selected for K-means, it can only detect the minority of the abnormal logs.

Varied from K-means, **K-means Dist** is designed as that uses K-means to cluster data into k clusters, and then map these k clusters into 2 categories (i.e abnormal and normal logs). We compute the distance between one instance and its nearest centroid, and those with values greater than threshold θ were considered as abnormal instances. Because in K-means, each instance will be assigned to one cluster, but those normal logs in one cluster usually are concentrated.

Figure 5.3 shows the distances between points to their centroids of the sample data set using K-means Dist with proper k. Figure 5.4 is the visualization of the clustering results.

The result shows that it is more effective than basic K-means with $k = 2$, but this method needs to select proper k through the cost function plot which is not applicable in the real implementation. Although we can iterate some values and select the best one for one model, it cannot be applied directly to other situations.

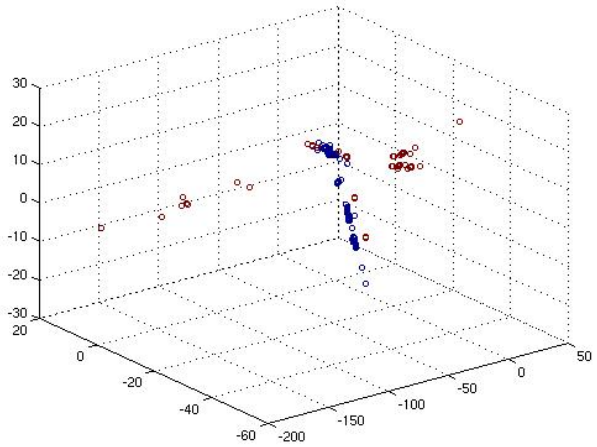


Figure 5.1: The verdicts of the testing results of the sample data set.

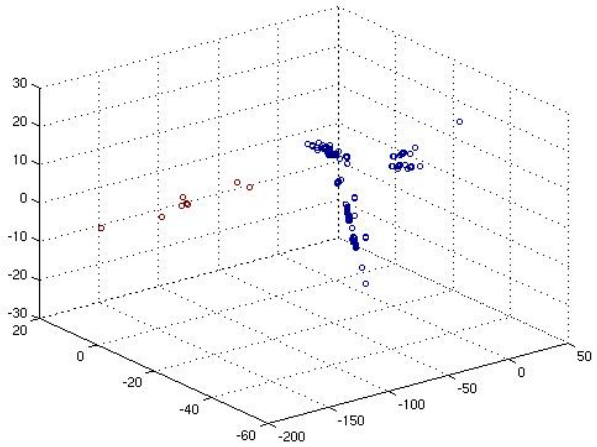


Figure 5.2: Clustering results of the sample data set using K-means with $k = 2$

5.1.2 DBSCAN Variants

DBSCAN (*for density-based spatial clustering of applications with noise*) [9] is another simple but effective clustering algorithm and it is based on density of the data. A common way human used to identify clusters is to count the points in a region, and if a region includes obviously more points than other places, this region forms a cluster of these points in this region. DBSCAN adopts a similar approach.

Firstly, some terms are introduced.

- **core point** If the number of points in one point p's neighborhood is greater than the threshold $MinPts$, p is a core point.
- **border point** The border point is not a core point, but it is located in one or multiple core points' neighborhoods.
- **outlier** (noise point) The other points except the core points and border points are all outliers.

A common DBSCAN algorithm will exclude outliers in the final clustering results. Since the outliers are interesting for this project, an algorithm called **DBSCOD** (*for density-based spatial clustering of outlier detection*) is varied from DBSCAN. It keeps these outliers and let them be clustered into one cluster in which the instances were predicted as abnormal instances.

It is not suitable to deal with high dimensionality data. Figure 5.5 shows the visualization of the clustering results of the sample data set using DBSCOD.

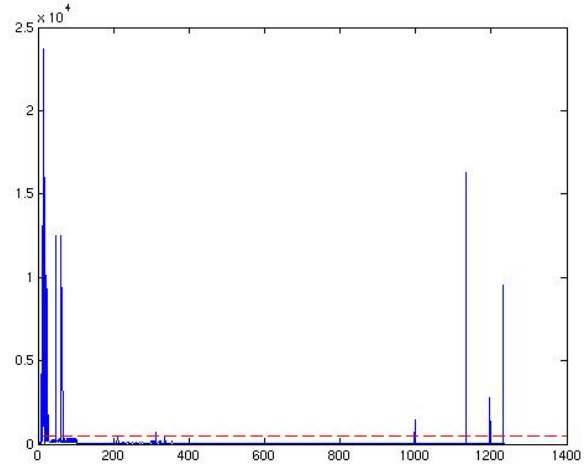


Figure 5.3: Distances plot of the sample data set with proper k

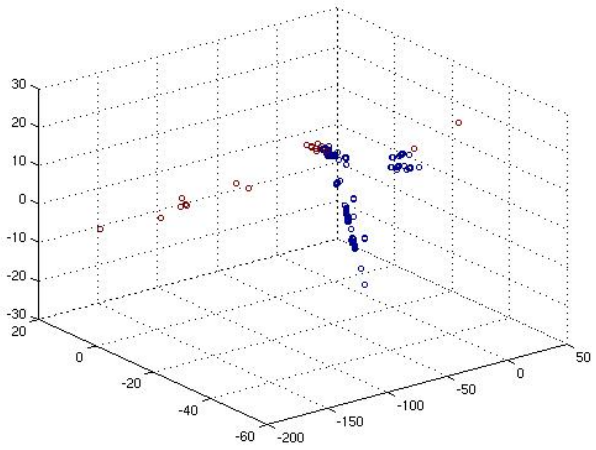


Figure 5.4: Clustering results of the sample data set using K-means Dist with the same k

5.1.3 SOFM Variants

SOFM (*self-organizing feature map*) or **SOM** (*self-organizing map*) [16] is an effective clustering algorithm in artificial neural network (*ANN*) [8]. From another perspective, SOFM is viewed as an kind of clustering algorithm based on centroids as K-means, and it assigns each data point to the most similar centroid. But it is different that SOFM imposes the centroids a topographic organization and the updating process of centroids is different. Next, the details of SOFM's characteristics will be introduced.

Firstly, some terminologies in SOFM should be explained.

- **neuron** Neurons are the nodes in the artificial neuron network, including input neurons and output neurons.
- **input neuron** Input neurons are these nodes in the input layer, that is, input neurons refer to these input data points.
- **output neuron** Output neurons are these nodes in the output layer (i.e. competitive layer). Each centroid is associated with an output neuron.
- **reference vector** A reference vector in the input space is represented as the weights of the connections from the input neurons to an output neuron.
- **BMU** (*the best matching unit*) A BMU for one data point p is an output neuron whose weight vector is most similar to p.

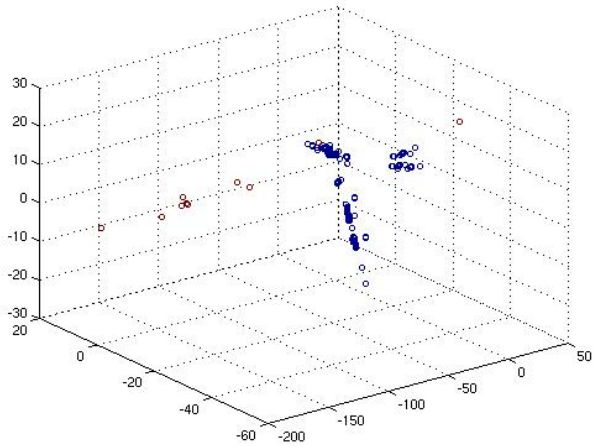


Figure 5.5: Clustering results of the sample data set using DBSCOD

SOFM, as an ANN, has two layers, *input layer* and *output layer* (i.e. *competitive layer*). The map topology of the competitive layer could be in many structures, e.g. a hexagonal grid, a triangular grid. Figure 5.6 is an example of a 5-by-5 hexagonal grid. The output neurons in the competitive layer are trained with an extended competitive learning method. As for each input neuron p, it is assigned to the closest output neuron u, which u is also called the winner. During the updating iterations, the neuron u is updated to be more likely to win next time, that is, the weight vector of u is moved towards p. Also, the close neighbors of the winner u are also adopted in different strengths dependent on their distances from u.

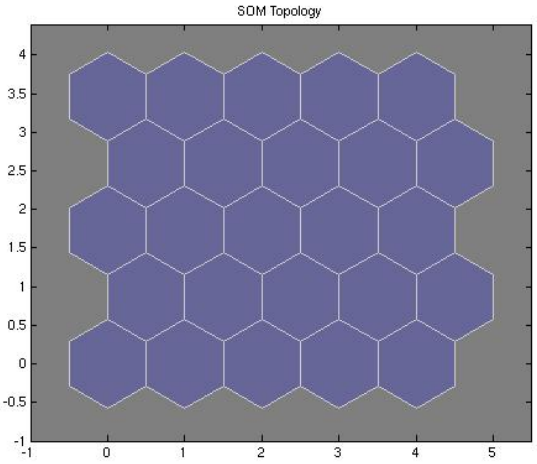


Figure 5.6: Illustration of a 5-by-5 hexagonal SOFM neuron network

The mathematical expression of the updating process is described below. In the time t, let $p_i(t)$ be the current input neuron, u_k be the winner, and $w_j(t)$ be the current weight vector of the p_i to its closest neuron u_j. In the time $t+1$, the weight of the output neuron u_j is updated as followed.

$$w_j(t+1) = w_j(t) + \eta f(u_j, u_k)(p_i(t) - w_j(t)) \tag{5.2}$$

η is the learning step which represents the learning rate with $0 < \eta < 1$ and decreases with the time going. $f(u_j, u_k)$ is the neighbourhood function which determines the updating strengths of the neighbors u_j of the winner u_k. It decreases with the increasing distances on the grid from the winner. $f(u_j, u_k)$ uses Gaussian function usually.

SOFM is often trained in two phases. The first phase is a coarse ordering phase, which starts with large neighbour radius and learning step and tries to find the number of categories. The next phase is a convergence phase (or called *tuning phase*), which starts with small neighbour radius and low learning step and tries to decide the exact locations of the neurons and the forms of the map [8].

Although there perhaps exists some empty clusters, the number of non-empty output clusters of SOFM is usually more than the number of actual categories, and a method should be found to map the output clusters into 2 categories (i.e. abnormal and normal logs).

SOFM-COUNT is based on the assumption that the number of abnormal logs is much less than the number of normal logs. It makes sense if we treat those clusters with seldom instances as abnormal clusters.

Take the sample data set as illustration. Figure 5.7 plots a SOFM layer, and the number of data points is shown within their BMU. The points whose BMUs are located in the red circles are viewed as abnormal logs. Figure 5.8 is the clustering results. Many normal logs are wrongly predicted as abnormal logs.

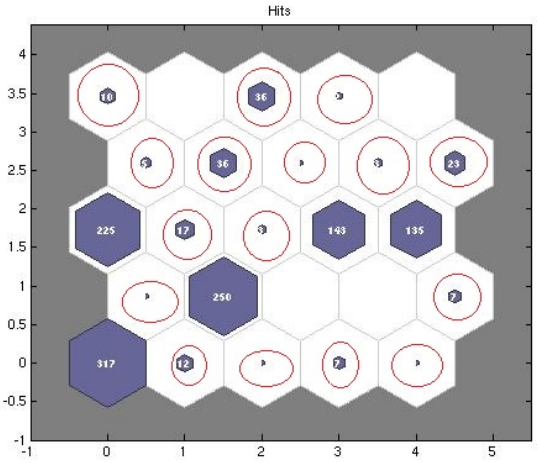


Figure 5.7: The sample hits plot of the sample data set using SOFM-COUNT

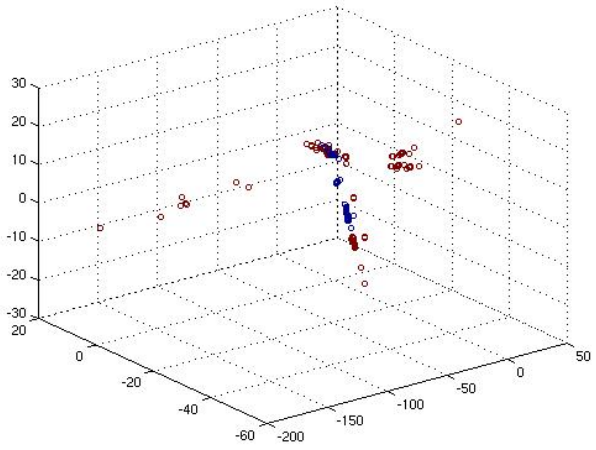


Figure 5.8: Clustering results of the sample data set using SOFM-COUNT

SOFM-DIST is inspired by K-means, that is, the distance between one data point and the weight vector of its BMU is computed, and those with exceptionally high values are considered as abnormal instances.

Figure 5.9 shows the distances between points to their BMUs of the sample data set using SOFM-DIST. Figure 5.10 is the visualization of the clustering results.

SOFM-DIST is more effective than SOFM-COUNT.

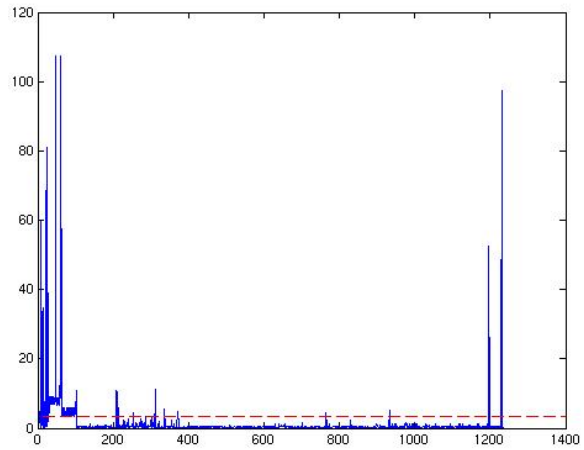


Figure 5.9: The distance plot of the sample data set using SOFM-DIST

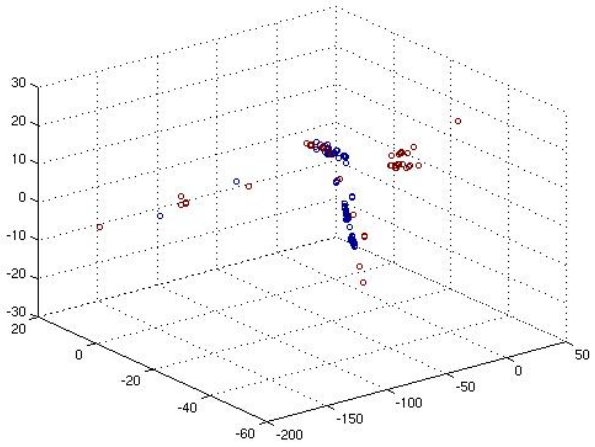


Figure 5.10: Clustering results of the sample data set using SOFM-DIST

5.2 Combined Algorithms

The following algorithms are inspired by the above basic algorithms and try to complement each other through the algorithm combination.

5.2.1 SOFM-Kmeans

SOFM-Kmeans is the first re-clustering method of our investigation on clustering the results of SOFM by other clustering algorithm, e.g. K-means with $k = 2$. The idea is to cluster the BMUs after SOFM into two clusters by K-means with $k = 2$. And the cluster with smaller number of data points is viewed as abnormal cluster.

The K-means results of BMUs see Figure 5.11. The BMUs in the red circle belong to one cluster. The clustering results of all data points see Figure 5.12.

The result is not good.

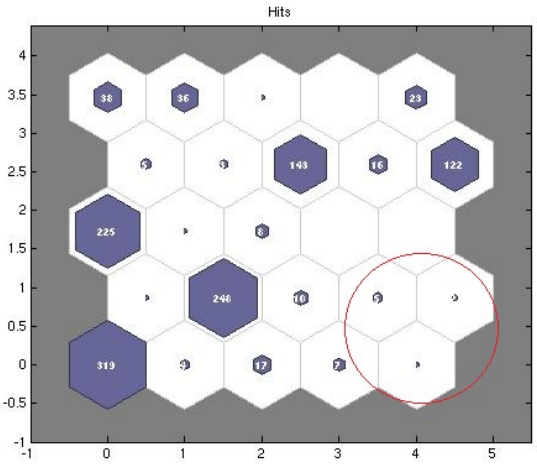


Figure 5.11: The K-means results of BMUs on the sample data set with $k = 2$.

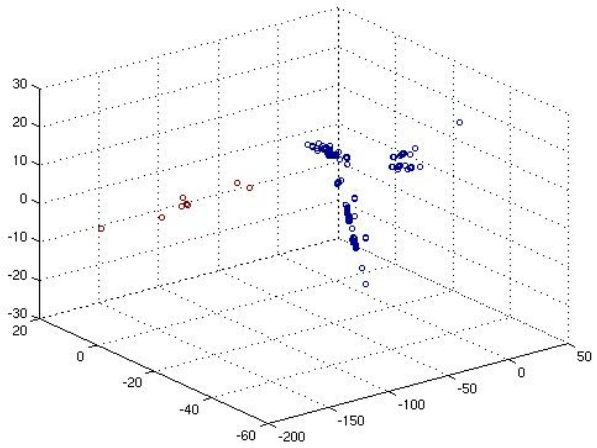


Figure 5.12: The clustering results of the sample data set using SOFM-Kmeans with $k = 2$.

5.2.2 Multilayer-SOFM

We tried to do boosting algorithms of SOFM, but it is found that boosting algorithms are mostly used for supervised learning algorithms due to the requirements of true answers to adjust the weights of different classifiers. As our case is unsupervised learning, we tries to produce a SOFM ensemble according to some priori knowledge and assumptions. **Multilayer-SOFM** uses SOFM as filters to filter the large clusters of the upper layer of SOFM and repeats fixed times or until there are not greater than two non-empty clusters. In the experiments, two filtering methods are used. One is to filter the data points in the largest one cluster and keep the other points into the next layer SOFM; the other way is to filter the points in the largest two clusters and pass the other points to the next layer SOFM. The number of layers is dependent on the data set.

The application of this algorithm is limited due to the restriction of priori knowledge and the assumptions. The parameter *numLayers* and *rules* are deeply dependent on the characteristics of the data points and these assumptions may not be established for other data sets because the distribution of anomalies is hard to predict in advance.

5.2.3 DSD/DSC

DSD(*DBSCOD-SOFMDIST*) uses DBSCOD as a filter and clusters the outliers in DBSCOD into two categories by SOFM-DIST. DSC (*DBSCOD-SOFMCOUNT*) is another variant that clusters the outlier in DBSCOD into two categories by SOFM-COUNT. These two methods are effective when DBSCOD predicts all abnormal logs correctly but tends to classify a log of normal logs as anomalies as well.

5.2.4 SOFM-DBSCAN

Detecting the outliers in the SOFM results was investigated as well. In SOFM-DBSCAN, SOFM is used to cluster log files firstly, and then DBSCOD is executed on their BMUs. For those data points whose BMUs are outliers predicted by DBSCOD are viewed as abnormal logs.

Figure 5.13 shows the distances between neighbor output neurons (i.e. gray-blue patches). The neighbor patches with darker color mean longer distances between these neurons. Figure 5.14 is the sample hits plot, in which the points whose BMUs are in the red circles are predicted as anomalies. Figure 5.15 is the clustering result of the sample data set using SOFM-DBSCAN. We can compare to Figure 5.1. SOFM-DBSCAN gets a very good result in the sample data set.

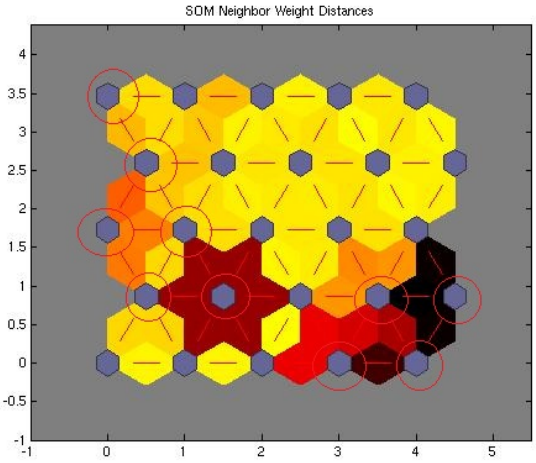


Figure 5.13: The neighbor weight distance plot of BMUs on the sample data set

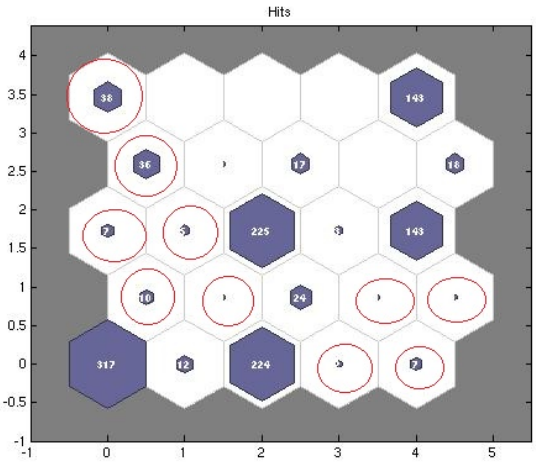


Figure 5.14: The neighbor weight distance plot of BMUs on the sample data set

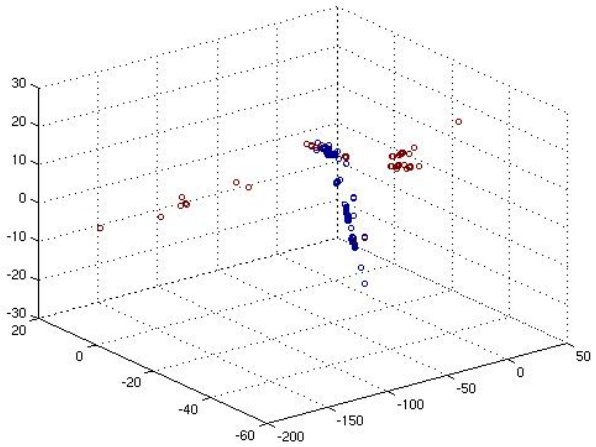


Figure 5.15: The clustering results of the sample data set using SOFM-DBSCAN.

6. Results

In this chapter, the results of different case studies are introduced. And from these case studies, the effects of different factors are concluded and different methods will be evaluated.

6.1 Metrics

In the section 3.1, we know that the problem in this project is a class imbalance problem and *accuracy* and *error rate* are not well suited for the evaluation of this kind of problem. Instead, **F-score** [33] is used to evaluate our results. Here, a brief introduction about the F-score and related concepts are presented.

Usually, the prediction results will be summarized in a table called **confusion matrix** [32]. The following terminologies in this table are interpreted using our problem and presented in Figure 6.1:

- True Positive (TP), which corresponds to the number of abnormal logs correctly predicted by the learning method.
- False Negative (FN), which corresponds to the number of abnormal logs wrongly predicted as normal logs by the learning method.
- False positive (FP), which corresponds to the number of normal logs wrongly predicted as abnormal logs by the learning method.
- True negative (TN), which corresponds to the number of normal logs correctly predicted by the learning method.

		Prediction Outcome	
		Abnormal	Normal
Actual Value	Abnormal	TP	FN
	Normal	FP	TN

Figure 6.1: Confusion Matrix

Recall and **precision** are two widely used metrics employed in applications of class imbalance problem [33]. Recall measures the proportion of the abnormal logs that are correctly predicted by the learning method. The learning method with high recall rarely mispredicts abnormal logs as normal logs. Precision measures the proportion of the real abnormal logs in those predicted abnormal logs by the learning method. The learning method with high precision rarely mispredicts normal logs as abnormal logs. Their definitions are given below.

$$\text{Recall}, r = \frac{TP}{TP+FN}, r \in [0,1] \tag{6.1}$$

$$\text{Precision}, p = \frac{TP}{TP+FP}, p \in [0,1] \tag{6.2}$$

However, only maximizing one of them is not enough. If a model predicts all logs as abnormal logs, it has a perfect recall but a terrible precision; on the contrary, if a model has high precision but low recall, all predicted logs are correct but there are many logs unpredictable. Therefore, a model that maximizes both recall and precision is needed to be generated.

F-score represents a metric combined from recall and precision and a model with high F-score ensures that the recall and precision are both high enough. The equation of F-score is listed as below:

$$\text{F-score}, F_1 = \frac{2rp}{r+p} = \frac{2 \times TP}{2 \times TP + FP + FN}, F_1 \in [0,1] \tag{6.3}$$

Those models with over 0.8 of F-score are treated as very good models; those between 0.7 and 0.8 are good enough; F-scores between 0.6 and 0.7 are fair; and those below 0.6 are a little poor. This interpretation may be different dependent on different problems.

False Alarm Rate (or *False Positive Rate*) [33] represents the ratio of normal logs wrongly predicted as abnormal logs by the learning method to the total normal logs:

$$\text{False Alarm Rate}, FAR = \frac{FP}{FP + TN}, FAR \in [0,1] \tag{6.4}$$

Generally, a model with lower FAR is better. But in this project, it is only able to compare the results to the test system verdicts. And those failed test cases are assumed to be really failed which means those logs in failed test cases are abnormal logs but those logs in passed test cases could be abnormal logs as well. It is possible that some of those False Positive logs are actually abnormal logs in reality, True Positives. Therefore, those FAR below 0.1 are tolerant in our experiments but the value may be different in different cases.

6.2 Case Study

We did experiments with multiple data sets. In this section, the experiment results are summarized in 7 case studies. Firstly, the recommended solution for this project is presented. In the following case studies, some experiments are used to verify some ideas. Limitations of space prevent us from covering the results of all tried models, so some parts of our experiments are presented to show our results in each case study. For more detailed results, see [20].

6.2.1 Recommended Solution

We tried more than 100 models and at least 5 models produced good results in certain test cases. Most of them are using the variants of SOFM as the learning algorithms, so it is reasonable to conclude that SOFM variants are more effective the other basic algorithms including K-means and DBSCAN. Table 6.1 shows the best F-scores for each learning algorithm that has been experimented [1].

Table 6.1: *The best F-scores of each learning algorithm in our experiment.*

Learning Algorithm	F-score
K-means (k=2)	0.141
K-means Dist	0.286
DBSCOD	0.175
SOFM-COUNT	0.648
SOFM-DIST	0.908
SOFM-Kmeans	0.144
Multilayer-SOFM	0.899
DSD/DSC	0.822
SOFM-DBSCAN	0.950

And because our evaluation is based on the verdicts of the test system which is not reliable enough, it is very hard to decide the best solution for our problem. When some method did not get a good result on certain

[1] This result only contains the results from the data set with more than 1000 log files and these F-scores could be in different data sets with different data preprocessing and extracted features.

test cases but got an almost perfect F-score in other test cases, it is hard to say that it performs bad generally. For example, Figure 6.2 is the results on different test cases of SOFM-DBSCAN with preprocessed TF-IDF and timestamp statistics as features.

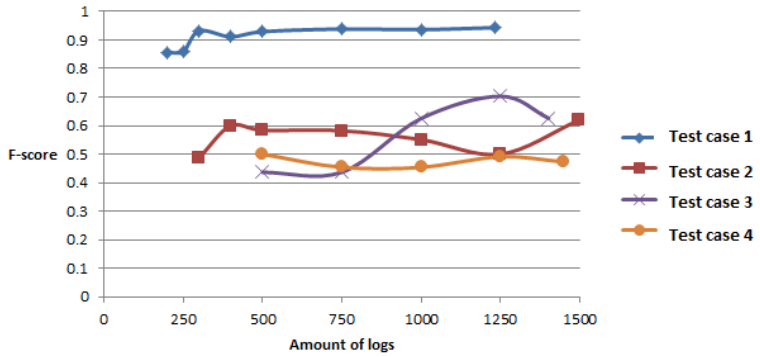


Figure 6.2: Results on different test cases of SOFM-DBSCAN+preprocessed TF-IDF+timestamp statistics.

Although it got a very high F-score of 0.943 in Test case 1, all the other three test cases produced poor F-scores. This solution is not a good choose for general test cases but it is very effective on certain test case. But, it is also possible that some of the passed logs in the other three test cases are actually abnormal logs and the solution can detect them. Then the F-score will be higher in that case.

If a suitable solution is necessary to be chosen for general cases, the recommended solution is using **character bigram** and **timestamp statistics** as features without text normalization and using **SOFM-DIST** as our learning method. The results on different test cases of this solution are in Figure 6.3.

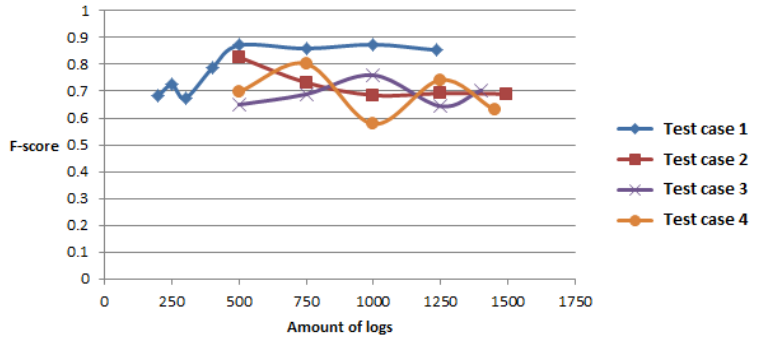


Figure 6.3: Results on different test cases of our recommended solution.

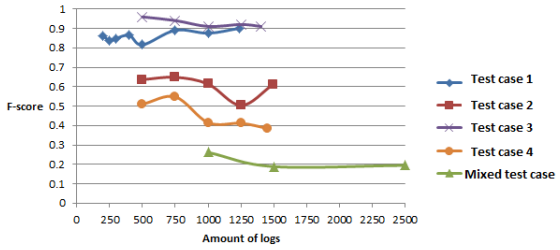
Figure 6.3 shows that this solution gives good results independent on different test cases while other solutions perform differently in different test cases. And also it has the potential ability to find unknown problems. Because our evaluation is based on the test system verdicts, if a method has a nearly perfect F-score, it means it did not find some false positive logs that may contain some unknown problems. Therefore, a solution with a good F-score, not a perfect F-score is chosen.

6.2.2 Different Test Cases

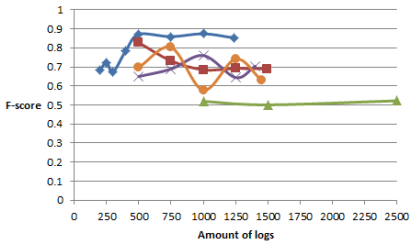
Here, **test case** is one possible factor that may raise problem 2, because different test cases may produce hugely different logs. We tested on multiple data sets with different single test case and one data set with mixed test cases under otherwise equal conditions. The assumption is:

Assumption 1 The data set with only one test case performs better than those with multiple test cases under otherwise equal conditions.

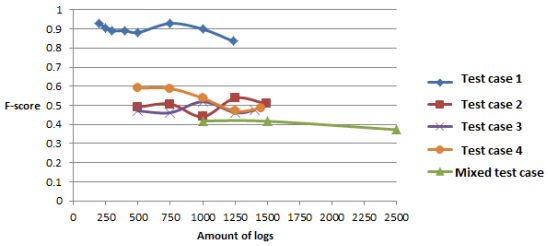
Figure 6.4 gives the results of the experiments in which character bigram and timestamp statistics are used as features without data preprocessing but with normalization and dimensionality reduction and use three kinds of learning algorithms, i.e. SOFM-DBSCAN, SOFM-DIST and Multilayer-SOFM.



(a) Results of SOFM-DBSCAN on different test cases



(b) Results of SOFM-DIST on different test cases



(c) Results of Multilayer-SOFM on different test cases

Figure 6.4: Parts of results on the influences of different test cases.

The results show that the F-scores of single test case are much higher than that of mixed test case independent on different learning methods. The results prove our Assumption 1.

6.2.3 Different Track Types

Next, **track type** is another possible factor that may raise problem 2, because the testing points in different track types may be very different. The assumption is:

Assumption 2 The data set with only one track type performs better than those with multiple track types under otherwise equal conditions.

We tested on one data set with single track type and one data set with mixed track types. They are all from the same test case (Test case 1). The results of experiments with preprocessed TF-IDF and timestamp statistics as features and SOFM-DBSCAN as learning method are in Figure 6.5. And because the number of logs of the data set of single track type is few, multiple data sets are tested without certain track type instead of with single track type. The results of the same solution are in Table 6.2.

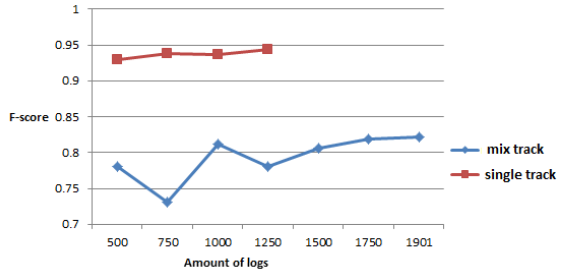


Figure 6.5: Results of single and mixed track types.

Figure 6.5 shows that the F-score of single track type is much higher than that of mixed track types. In Table 6.2, F-score becomes higher if there are few track types and it reaches 0.943 when there is only one track type, which proves our Assumption 2 correct indirectly.

Table 6.2: *Results of data sets without certain track type.*

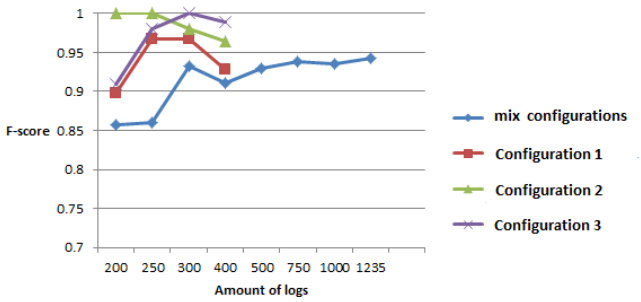
Test Case	Track	Amount	Recall	Precision	FAR	F-score
Test case 1	mix	1901	0.864	0.788	0.016	0.822
Test case 1	Exclude Track 1	1881	0.864	0.798	0.015	0.827
Test case 1	Exclude Track 2	1791	0.828	0.888	0.008	0.848
Test case 1	Exclude Track 3	1664	0.88	0.858	0.012	0.867
Test case 1	only Main Track	1235	0.923	0.974	0.003	0.943

6.2.4 Different Configuration Types

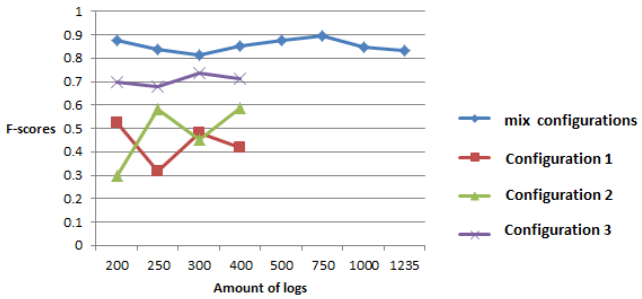
Configuration types also could be the factor that may raise problem 2, because the test scripts for different configuration types may be very different. The assumption is:

Assumption 3 The data set with only one configuration type performs better than those with multiple configuration types under otherwise equal conditions.

We tested on different configuration types with different solutions, and the results did not prove our Assumption 3 since different learning methods produced different results. The example given in this case study is using preprocessed TF-IDF and timestamp statistics as features with different learning methods on the sample data set. Figure 6.6a is the results of SOFM-DBSCAN and Figure 6.6b is the results of SOFM-DIST.



(a) Results of SOFM-DBSCAN on single and mixed configuration type.



(b) Results of SOFM-DIST on single and mixed configuration type.

Figure 6.6: Parts of results on the influences of different configuration types.

In Figure 6.6a, those data sets with single configuration type perform better than that with mixed configuration types. But in Figure 6.6b, the data set with mixed configuration types produces higher F-score. So it seems that the results are dependent on different learning methods. But due to limited number of logs in single configuration type, the results might be unstable and perhaps it will be different conclusion if there are more samples.

But anyway, the results of mixed configurations are seemed to be good enough. In Figure 6.6b, the experiment with the mixed configuration types got a F-score of 0.833 and in Figure 6.6a, the F-score reaches 0.943 which is really good. So it is not necessary to learn models on separate configuration types.

6.2.5 All Pass Test Case

In our data sets, there are many test cases containing all passed logs and no failed or error logs. If F-score is used to evaluate the clustering ability of our learning method, it is not fair as F-score always equals to zero. But because it is hard to trust the test system verdicts, some of the passed logs may be abnormal logs in fact. And those predicted abnormal logs by our learning method are highly suspicious. Therefore, one test case with all passed logs is tried. The test case is Test case 5 in which there are 1449 passed logs in total.

We extracted character bigram and timestamp statistics as features and did clustering by SOFM-DIST. Our learning system detected 13 abnormal logs which are highly suspicious. In the 13 logs, one log file is empty, one is very long compared to other normal logs and the other 11 logs all contain a suspicious string.

It is obvious that the former two log files are abnormal and the contents of all logs in this test case are analyzed. It shows that only these 11 logs contain such a string. In other words, our learning method detected all logs including this string among 1449 logs. So this kind of logs is very special compared to others.

With regard to our findings, we asked an expert and from his comments we know that this string seems to be able to cause some problems which should be investigated later. Therefore, this kind of logs contains abnormal information that our learning method detected successfully. We can see that our learning method is effective in detecting abnormal information in the logs from this case study.

6.2.6 With and Without Text Normalization

In the investigations on the effectiveness of text normalization, experiments on different solutions both with and without text normalization have been done. Experiment results are dependent on different kinds of features.

For example, it is necessary for TF-IDF to do text normalization since this kind of feature concentrates on the words. If there is no text normalization, the tokens in the logs are separated by space only and it is possible that there are some redundant characters combined with the same word so that it could be several TF-IDFs for the same word. It is the same for the other features on words (i.e word bigram, word count). Table 6.3 shows the same data set with and without text normalization for TF-IDF by different learning methods. We also test doing parts of the text normalization jobs and it shows that completing all operations is effective for TF-IDF. For detailed results, see [20].

Table 6.3: *Text normalization investigation of TF-IDF on the sample data set.*

Text Normalization	Algorithm	Recall	Precision	FAR	F-score
yes	SOFM-DBSCAN	0.923	0.974	0.003	0.943
no	SOFM-DBSCAN	0.761	0.2	0.289	0.313
yes	SOFM-DIST	0.894	0.79	0.024	0.833
no	SOFM-DIST	0.126	0.68	0.002	0.201

However, it is complicated for character bigram. In some cases, it is better to do without text normalization for character bigram. It is mainly due to that the unit for character bigram is each character and each character could be helpful for the prediction in those cases. If text normalization is done, perhaps some useful informations will be removed; in other cases, extracting character bigram from normalized log files is better, which may due to the deletion of the redundant characters in those log files. Table 6.4 records the results on different test cases with and without text normalization for character bigram by SOFM-DBSCAN.

Although TF-IDF, word bigram or word count were not chosen as our features, they could be investigated with other kinds of text normalization as well in the future. For example, remove all simple words (such as a, the), and extract stems of each word instead that different forms of one word are treated as different words.

Table 6.4: *Text normalization investigation of character bigram on the sample data set.*

Text Normalization	Recall	Precision	FAR	F-score
yes	0.923	0.919	0.008	0.919
no	0.922	0.902	0.01	0.91
yes	0.791	0.437	0.061	0.549
no	0.65	0.655	0.02	0.626
yes	0.86	0.819	0.015	0.8
no	0.933	0.917	0.004	0.918
yes	0.752	0.837	0.011	0.761
no	0.862	0.337	0.157	0.419

6.2.7 Different ways to do Dimensionality Reduction

When different features are combined together, there could be more than 10000 dimensions in the feature matrix. Each kind of feature can contain thousands of attributes. In order to avoid dimensionality disaster mentioned in section 3.2.3, it is necessary to do dimensionality reduction (*DR*) in this project. But there are different ways to do that with combined features. Way 1 is to reduce the dimensions of each kind of feature separately and then merge them together as a big feature matrix. Way 2 is to merge different features together firstly and then reduce the dimensions of the whole feature matrix. We use experiments to tell which way is proper for our problem. Table 6.5 shows results of different feature combinations and learning methods by different ways of dimensionality reduction on the same data set.

Table 6.5: *Dimensionality reduction investigation on the same data set.*

Feature	Algorithm	DR Way **	Recall	Precision	FAR	F-score
TF-IDF + time *	SOFM-DIST	1	0.97	0.446	0.075	0.602
TF-IDF+ time	SOFM-DIST	2	0.853	0.799	0.015	0.82
TF-IDF+ time	SOFM-COUNT	1	0.7	0.382	0.085	0.429
TF-IDF+ time	SOFM-COUNT	2	0.979	0.536	0.062	0.679
word bigram+ TF-IDF+ time	SOFM-DIST	1	0.959	0.246	0.196	0.389
word bigram+ TF-IDF+ time	SOFM-DIST	2	0.979	0.536	0.062	0.679

* time refers to timestamp statistics.

** 1 is to reduce the dimensions of each kind of feature separately and then merge them together as a big feature matrix. 2 is to merge different features together firstly and then reduce the dimensions of the whole feature matrix.

Our experiments prove that doing dimensionality reduction on the matrix of all kinds of features performs better than doing that separately on single kind of features and then combining together under otherwise equal conditions.

7. Conclusion and Discussion

This project's major findings and the improvements compared to AALA 1.0 are concluded in this chapter. Besides, the limitations and future work will be discussed in the following sections.

7.1 Conclusion

AALA 2.0 extends and improves the ideas of AALA 1.0 so that it is proved that it is possible to analyze logs automatically based on the features extracted from the contents and unsupervised learning algorithms. We also present some novel variant learning methods to solve our problem and prove that they are better than the original basic methods. With different case studies, the influences of types of different logs are investigated and different test cases is found to be possibly an important factor that affects the analysis results. Finally, the recommended solution (i.e. charbigram and timestamp statistics as features and SOFM-DIST as learning method) is described, which can detect almost all failed logs in the automated testing results and also find unknown problems, as a good solution for general cases.

7.2 Major Findings

Firstly, from the experiment results, training one model per test case is preferred in this project. The case studies in section 6.2 show that the model based on single test case performs better than those based on multiple test cases under otherwise equal conditions and the model based on only one track type performs better than those based on multiple track types under otherwise equal conditions. But the preliminary results of different configuration types with limited logs show that it depends on different learning methods. Since the results of mixed configuration types seem to be good enough, it is not necessary to train one model per configuration type. But it makes harder to detect abnormal logs if we learn on mixed test cases. Therefore, it is suggested to training one model per test case in the later implementation phase.

Secondly, among those feature candidates, the results show that simple features such as character bigram and timestamp statistics are effective enough to distinguish abnormal and normal logs. The advanced feature TF-IDF is more effective in certain test case, but gets fair results in some other test cases. It is better to investigate whether it is because those test system verdicts are not reliable or TF-IDF is not effective in certain cases in the future.

Among those learning algorithm candidates, some effective variants of SOFM are developed and the results are better than basic clustering algorithms like K-means and DBSCAN. And these variants solved the problem of matching multiple clusters into two categories (i.e. abnormal and normal) of the basic SOFM.

It is hard to select the best solution for all test cases since the evaluation is based on the test system verdicts and it could also be that different test cases have different best solutions. If it is necessary to select a suitable solution for general cases, the combination of character bigram and timestamp statistics is chosen as the features and SOFM-DIST is chosen as the learning algorithm. They are thought to be good independent on different test cases.

Thirdly, the experiment results show that dimensionality reduction is necessary for this project. The possible reason is that our data is a high-dimensional data without dimensionality reduction and is located in a sparse space. It makes harder for the clustering algorithms if the definition of density and distance between two data points becomes meaningless. Dimensionality reduction is also helpful to remove the irrelevant feature terms and reduce the noises. Also, dimensionality reduction can increase efficiency and

decrease the consuming of time and memory resources. So it is necessary for our problem. And the experiment results also prove that doing PCA after combinations of features is better than doing PCA on individual feature and then merging them together.

Additionally, different sizes of data sets are tested and it is found that it is better to have more than 1000 training samples in order to learn a model with stable results. When there are few training samples, the whole data set might be in a low density space which is not good for clustering and few training samples cannot represent the diversity of the log files well. But if there are too many training samples, then it may lead to over-fitting problem, so it is suggested to have 1000~1500 logs for our problem.

Finally, someone with some knowledges on testing tried to look at the parts of our logs and there are many failed logs that human cannot find out but our machine can detect. Also from the result of the ALL-PASS Test case 5, we can see that the machine learning solution in this project is effective in detecting abnormal information in the logs that test system and even humans may not notice.

7.3 Improvements VS AALA 1.0

Due to limited time, there are many shortcomings in AALA 1.0 and AALA 2.0 improves majority of them.

Firstly, more effective learning methods are investigated and developed, for example, SOFM-DIST, SOFM-COUNT, SOFM-DBSCAN and Multilayer-SOFM. In AALA 1.0, only basic K-means, DBSCAN and PCA-based anomaly detection were tested. And it did not solve the parameter selection problem and cluster matching problems of those algorithms. Those variants in AALA 2.0 can choose suitable parameters via iterations and solve the cluster matching problems,

Secondly, more than 100 combinations of features and algorithms were tested in AALA 2.0. In AALA 1.0, the author put all features together as the final feature matrix. We have more feature candidates and learning algorithm candidates than AALA 1.0 and different combinations of features are tested instead of putting them together as a single feature matrix, and different matches of feature combinations and learning algorithms are tested.

Thirdly, there is a more systematic learning process in AALA 2.0. We select a suitable metric to do evaluation. In AALA 1.0, the author did not use a mathematical metric to evaluate the results which made the evaluation ambiguous. Also, cross validation was performed in AALA 2.0. In AALA 1.0, the training and testing are on the same data set, which may lead to over-fitting problems and the learnt model cannot guarantee a good generalization ability on new data sets. Cross validation is a common technique used in machine learning problems.

Additionally, more data sets were investigated in AALA 2.0. Due to limited time, AALA 1.0 only tested on one data set with two types of logs. In AALA 2.0, multiple data sets with different compositions of log files were tested. And we propose and verified that different test cases, track types may disturb the learning results through experiments on different data sets.

Last but not least, based on more effective methods, results in AALA 2.0 are much better than that in AALA 1.0. In AALA 1.0, the best result is using K-means with character bigram, timestamp statistics and word count features and it found one suspicious log that should be abnormal but passed in the test system. In AALA 2.0, F-score is used to evaluate our results and the best one can be over 0.9, which shows that the learning method can detect majority of those failed logs in the test system. If the method in AALA 1.0 is used to test on our data set, the F-score is less than 0.4. And the results of the ALL-PASS test case also shows that AALA 2.0 can detect those suspicious logs that should be abnormal but passed in the test system.

7.4 Limitations

The biggest limitation in our project is that it is hard to find experts to analyze the logs manually. Although the clustering algorithms do not need any feedback during learning, the evaluation of different models need

to compare our prediction results with the true answers. But it is not possible to find the experts to do manual log analysis in this project, therefore, the evaluation is based on the test system verdicts. The test system verdicts are not 100% correct. We assume that those failed test cases can be trusted but those passed test cases may not really pass. Therefore, those False Positive logs predicted by our learning method may not be predicted wrongly. It is possible that those logs are really abnormal but the test system did not find them out. Therefore, if the experts are available, it will be interesting to analyze those False Positive logs to see if they are in fact True Positive logs. It can not only correct the results but also can show the ability of finding unknown errors of our learning method. Besides, the experts may help us find better features with professional knowledge on log analysis.

Another limitation is that some interesting logs are limited in our experiments. The number of logs of single configuration in single test case is less than 500, which may lead to incorrect conclusions about the effect of different configuration types. It will be better if more samples can be investigated. And many test cases in the experiments only contain pass logs. It is hard to do good evaluations on these test cases as the F-score always equals to zero for different methods. If there are more failed samples in these test cases, we can see the results of different solutions on those test cases.

Finally, because our evaluation is based on the verdicts of test system which are not reliable enough, it is hard for us to select the best model. Although some methods can get very good F-scores on certain test cases, they did not perform good on all test cases. However, the reason might be that those test cases contain more suspicious logs that should be abnormal but passed in the test system, therefore, it is not good to say that these methods perform bad. If reliable answers of these logs are available, it is easier to select the best model.

7.5 Future Work

There are much work that can be done in this project in the future. If it is possible to find real experts, those False Positive logs can be analyzed to check their abnormalities. It may improve results, help us to select the best model and prove the ability of finding unknown problems of our learning method. Also, the algorithms in this project can be tuned better based on correct results. With experts, it is possible to find more effective features as well. Different lengths of N-gram could be tested in the future. And if there are more failed logs in these ALL-PASS test cases, the solutions in this project can be investigated on those test cases.

If the experts are not available, perhaps we can evaluate our models from another perspective. For example, the correlation between the pass rate of different models in the previous test loop and the pass rate in later test loops can be investigated, that is, if the pass rate of one model in the previous test loop is low which may stand for the quality of the system is not good enough, and it may indicate the later loops have a risk of failing. A model is considered as good if it can detect this kind of correlation between the previous test loop and later loops.

Additionally, implementing on-line learning can be one future work, which means the system can improve itself by getting feedbacks. If human finds the result of the learning system is not correct, then gives feedback to the system. It can learn a better model next time. And other ways to do text normalization can be tried and TF-IDF, word bigram and word count can be tested again.

Also, the recommended solution should be very generic and could be applied to other types of logs. This could be tested on other logs in the future.

Finally, it is necessary to implement the algorithms in this project into a standalone service that can be used in other applications.

References

[1] M. Aharon, G. Barash, I. Cohen, and E. Mordechai, "One graph is worth a thousand logs: Uncovering hidden structures in massive system event logs," *Machine Learning and Knowledge Discovery in Databases*, 2009.

[2] D. Barbara, N. Wu, and S. Jajodia, "Detecting novel network intrusions using bayes estimators," in *First SIAM Conference on Data Mining*. Citeseer, 2001.

[3] P. Brucker, "On the complexity of clustering problems," *Optimization and Operations Research*, 1977.

[4] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," *Ann Arbor MI*, vol. 48113, no. 2, 1994.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[6] R. Dunia and S. J. Qin, "Multi-dimensional fault diagnosis using a subspace approach," in *American Control Conference*. Citeseer, 1997.

[7] V. Emamian, M. Kaveh, and A. H. Tewfik, "Robust clustering of acoustic emission signals using the Kohonen network," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 6. IEEE, 2000.

[8] A. P. Engelbrecht, *Computational intelligence: an introduction*. Wiley, 2007.

[9] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." Kdd, 1996.

[10] Y. Guan, A. A. Ghorbani, and N. Belacel, "Y-means: A clustering method for intrusion detection," 2003.

[11] A. Hilmkil, "Awesome automated log analysis," 2012, date accessed 2012-08-25. [Online]. Available: http://cdmweb.ericsson.se/TeamCenter/controller/ViewDocs?DocumentName=EAB%2FFJW-12%3A1027&Latest=true

[12] F. V. Jensen, *An introduction to Bayesian networks*. UCL press London, 1996, vol. 74.

[13] H. Jiawei and M. Kamber, "Data mining: concepts and techniques," *San Francisco, CA, itd: Morgan Kaufmann*, vol. 5, 2001.

[14] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[15] D. Jurafsky and J. Martin, *Speech & Language Processing*. Pearson Education India, 2000.

[16] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[17] P. Kumpulainen and K. Hätönen, "Local anomaly detection for network system log monitoring," in *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks*, 2007.

[18] K. Labib and R. Vemuri, "NSOM: A real-time network-based intrusion detection system using self-organizing maps," *Networks and Security*, 2002.

[19] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4. ACM, 2004.

[20] W. Li, "Experiment results of aala 2.0," 2013, date accessed 2013-05-25. [Online]. Available: https://eforge.ericsson.se/sf/go/doc200723

[21] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[22] C. Lonvick, "The bsd syslog protocol," 2001.

[23] G. McLachlan, K.-A. Do, and C. Ambroise, *Analyzing microarray gene expression data*. Wiley-Interscience, 2005, vol. 422.

[24] T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, 1997.

[25] S. Nousiainen, J. Kilpi, P. Silvonen, and M. Hiirsalmi, "Anomaly detection from server log data."

[26] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE, 2007.

[27] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.
[28] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, “A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases,” *International Journal of Computer Applications IJCA*, vol. 3, no. 6, 2010.
[29] M. Ramadas, S. Ostermann, and B. Tjaden, “Detecting anomalous network traffic with self-organizing maps,” in *Recent Advances in Intrusion Detection*. Springer, 2003.
[30] T. Sipola, A. Juvonen, and J. Lehtonen, “Anomaly Detection from Network Logs Using Diffusion Maps,” *Engineering Applications of Neural Networks*, 2011.
[31] P. Spyns, “Natural language processing,” *Methods of information in medicine*, vol. 35, no. 4, pp. 285–301, 1996.
[32] S. V. Stehman, “Selecting and interpreting measures of thematic classification accuracy,” *Remote sensing of Environment*, vol. 62, no. 1, pp. 77–89, 1997.
[33] P. N. Tan, M. Steinbach, V. Kumar, and Others, *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.
[34] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag New York Inc, 2000.
[35] Wikipedia, “Continuous integration,” 2012, date accessed 2012-10-01. [Online]. Available: http://en.wikipedia.org/wiki/Continuous_integration
[36] ——, “Radio base station,” 2012, date accessed 2012-10-01. [Online]. Available: http://en.wikipedia.org/wiki/Radio_Base_Station
[37] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. (2009, Oct.) Detecting Large-Scale System Problems by Mining Console Logs. ACM. Date accessed 2012-08-16. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.7257
[38] C. Yuan, N. Lao, J. R. Wen, J. Li, Z. Zhang, Y. M. Wang, and W. Y. Ma, “Automated known problem diagnosis with event traces,” in *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4. ACM, 2006.
[39] G. U. Yule, *An introduction to the theory of statistics*. C. Griffin and company, limited, 1919.
[40] J. Zheng, M. Hu, B. Fang, and H. Zhang, “Anomaly detection using fast sofm,” in *Grid and Cooperative Computing-GCC 2004 Workshops*. Springer, 2004.