
**ADDIS ABABA UNIVERSITY**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY**

**SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING**

# Alarm Prediction for Fault Management using Deep Learning Approach: The case of Ethio Telecom

By: Betelhem Berhanu

Advisor: Dr. Rosa Tsegaye

**A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Telecommunications Engineering.**

Addis Ababa, Ethiopia

September, 2021

**ADDIS ABABA UNIVERSITY**

**ADDIS ABABA INSTITUTE OF TECHNOLOGY**

**SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING**

# Alarm Prediction for Fault Management using Deep Learning Approach: The case of Ethio Telecom

Submitted by: Betelhem Berhanu Signature_____

Dr. Rosa Tsegaye
Advisor

_____
Signature

_____
Evaluator

_____
Signature

_____
Evaluator

_____
Signature

# Declaration

I certify that this Master of Science in Telecommunications Engineering thesis was

i. written wholly by myself,

ii. is the result of my own efforts, and

iii. has not been submitted for any other degree or professional qualification.

Betelhem Berhanu _____

Name Signature

Place: Addis Ababa, Ethiopia

Date of Submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Rosa Tsegaye _____

Advisor Name Signature

# Acknowledgment

I would like to thank first and foremost the almighty God for his never-ending grace. Many thanks, to my advisor Dr. Rosa Tsegaye for her continuous support throughout this thesis. I would also like to thank my family. Without them, this thesis wouldn't be successful. Finally, I would like to thank NNOC staffs who have supported me in data collection and technical guidance.

# Abstract

Telecommunication networks play a critical role in our society. They make it possible to share a massive amount of data across the globe. While networks are complex systems in terms of size and technological diversity, a failure results in numerous alarms from the series of devices. This makes monitoring and maintenance activities challenging due to the growing complexity of alarm management systems and the need for highly educated experts to deploy.

Alarms are generated in vast quantities every day by today's large and complicated telecom networks. The alarm sequence offers significant information on the network's activity, but most of it is fragmented and hidden in the massive amount of data. Alarm regularities can be utilized in fault management systems, for example, to filter redundant alarms, locate network problems, and even anticipate catastrophic faults. In the presence of flooding alarms, alarms that are inadequately configured and maintained, and a large number of nuisance alarms, operators are expected to make vital judgments. If the incoming alarms can be correctly predicted before they occur, the operators may be able to address and possibly avoid anomalous behaviors by taking corrective actions in a timely manner.

This paper presents an alarm prediction method based on data mining to generate patterns from historical alarm data, and use such patterns to train three deep learning approaches, namely long short-term memory (LSTM), bidirectional LSTM (Bi-LSTM) and gated recurrent unit (GRU). The prediction performance of the three deep learning approaches has been compared. Domain trained word embedding and pretrained word embedding (Word2vec) are used to feed embeddings to neural networks. The relevance of applying different word embedding is to explore the effect of the data preparation on the model performance. The Frequent pattern growth (FP) algorithm implemented in Rapidminer studio has been used to mine five months' worth of alarm logs. Finally, the best performing model is selected based on the accuracy of the model. The models are tested with a sequence of alarms and Bi-LSTM with domain trained word embedding achieves 93% in predicting the target alarms. However, from the results, we can also say that all three deep learning approaches can be used for predicting telecom alarms.



**Keywords**: - Data mining, Long Short Term Memory, Bidirectional LSTM, Gated Recurrent Unit, Word Embedding.

# Table of Contents

Declaration ..... ii

Acknowledgment ..... iii

Abstract ..... iv

List of Table ..... viii

List of Figure ..... ix

Abbreviations ..... x

1. Introduction ..... 1

1.1 Statement of the Problem ..... 3

1.2 Objective ..... 4

1.2.1 General Objective ..... 4

1.2.2 Specific Objectives ..... 4

1.3 Methodology ..... 5

1.4 Literature Review ..... 6

1.5 Scope and Limitation ..... 8

1.6 Contribution ..... 9

1.7 Thesis Organization ..... 9

2. Technical Background ..... 10

2.1 Fault Management in Telecom Networks ..... 10

2.1.1 Fault Detection ..... 10

2.1.2 Impact Analysis ..... 11

2.1.3 Recovery Action ..... 11
2.2 Alarms ..... 11
2.3 Data Mining ..... 13
2.3.1 Market Basket Analysis ..... 14
2.4 Association Rules ..... 15
2.5 Frequent pattern Mining algorithms ..... 17
2.5.1 Apriori ..... 17
2.5.2 Frequent pattern (FP) Growth ..... 18
2.6 Pattern Mining Tools ..... 19
2.7 Deep Learning ..... 20
2.7.1 Simple Recurrent Neural Network (Simple RNN) ..... 21
2.7.2 Long Short-Term Memory (LSTM) ..... 22
2.7.3 Gated Recurrent Unit (GRU) ..... 23
2.7.4 Bidirectional LSTM (Bi-LSTM) ..... 24
2.8 Optimization techniques ..... 25
3. Alarm Prediction Methodology ..... 27
3.1 Data Collection and Preprocessing ..... 28
3.1.1 Alarm Features ..... 28
3.1.2 Statistics of collected Data ..... 29
3.1.3 Data Cleaning ..... 31
3.1.4 Data Normalization ..... 32
3.2 Alarm pattern mining ..... 32

3.2.1 Matrix of transactions .....32
3.2.2 Rule generation.....34
3.2.3 Rule Visualization .....34
4. Model Training and Evaluation .....36
4.1 Deep Learning Model Training .....36
4.1.1 Hyperparameter setup .....37
4.1.2 Training using domain trained Word Embedding .....38
4.1.3 Training using Pretrained Word Embedding .....38
4.2 Model Evaluation .....38
5. Result and Discussion .....40
5.1 LSTM based model Result .....40
5.2 Bi-LSTM based model Result .....41
5.3 GRU based model Result .....42
5.4 Discussion .....42
6. Conclusion and Future Works .....44
6.1 Conclusion.....44
6.2 Future Works.....44
Reference.....46
Appendix A.....51

# List of Table

Table 1.1 Ethio telecom 6-month Incident report, 2020. Source: Adapted from [2]..... 3

Table 2.1 Example of BTS Hardware faults. Source: From [18].....12

Table 2.2 An example of market basket transactions Source: Adapted from [23] .....14

Table 2.3 A binary 0/1 representation of market basket data Source: Adapted from [23].....15

Table 3.1 Most common BTS alarms with their frequency of occurrence.....31

Table 3.2 An example of alarm transaction.....33

Table 3.3 A sample of association rule .....34

Table 4.1 Hyperparameter used .....37

Table 4.2 Test accuracy and loss with corresponding word embedding techniques .....39

Table 4.3 Test accuracy and loss with corresponding embedding dimension.....39

Table 4.4 Test accuracy and loss with corresponding batch size.....39

# List of Figure

Figure 1.1 Six month total average restoration time per domain, 2020. Source: Adapted from [2].... 3

Figure 1.2 Research methodology flow chart. ..... 6

Figure 2.1 The Fault Management Process. Source: Adapted from [14]. .....10

Figure 2.2 An itemset lattice Source: Adapted from [28].....18

Figure 2.3 Creation of Association rule in Rapidminer .....20

Figure 2.4 Simple RNN Source: Adapted from [35] .....21

Figure 2.5 Long Short-Term Memory Source: Adapted from [35].....23

Figure 2.6 Gated Recurrent Unit Source: Adapted from [35] .....24

Figure 2.7 Bidirectional LSTM Source: Adapted from [35] .....25

Figure 3.1 KDD process Source: Adapted from [40].....27

Figure 3.2 Alarms categorized by severity.....29

Figure 3.3 Alarms categorized by alarm types .....30

Figure 3.4 Alarms categorized by probable cause .....30

Figure 3.5 Alarm event versus date.....31

Figure 3.6 Scatter plot of rules.....35

Figure 5.1 LSTM model using word embedding learning curve.....40

Figure 5.2 LSTM model using pertained word embedding learning curve.....40

Figure 5.3 Bidirectional LSTM model using word embedding learning curve.....41

Figure 5.4 Bidirectional LSTM model using pertained word embedding learning curve.....41

Figure 5.5 GRU model using word embedding learning curve .....42

Figure 5.6 GRU model using pertained word embedding learning curve.....42

# Abbreviations

| | |
|---|---|
| AFSM | Acceptor Finite State Machine |
| ACO | Ant Colony Optimization |
| AI | Artificial Intelligence |
| BBU | Baseband Unit |
| Bi-LSTM | Bi-directional Long Short Term Memory |
| BTS | Base Transceiver Station |
| CPRI | Common Public Radio Interface |
| CRM | Customer Relationship Management |
| DM | Data Mining |
| DBN | Deep Belief Network |
| CNN | Convolutional Neural Network |
| DNN | Deep Neural Network |
| EMS | Element Management System |
| ET | Ethio Telecom |
| FMEA | Failure Modes and Effects Analysis |
| FMECA | Failure Modes, Effects and Criticality Analysis |
| FM | Fault Management |
| FMS | Fault Management System |
| FP | Frequent Pattern |
| GPU | Graphics Processing Unit |
| GRU | Gated Recurrent Units |
| GSM | Global System for Mobile |
| IBM | International Business Machines |
| IP | Internet Protocol |
| KDD | Knowledge Discovery in Databases |
| KPI | Key Performance Indicator |
| LHS | Left Hand Side |
| LSTM | Long Short Term Memory |
| LTE | Long Term Evolution |
| NNOC | National Network Operation Center |

| | |
|---|---|
| NLP | Natural Language Processing |
| NSN | Nokia Simmons |
| O&M | Operational and Maintenance |
| OFC | Optical Fiber Cable |
| QoS | Quality of Service |
| RAM | Random Access Memory |
| RNN | Recurrent Neural Network |
| ReLU | Rectified Linear Unit |
| RMS | Resource Management System |
| RF | Radio Frequency |
| RHS | Right Hand Side |
| SCTP | Stream Control Transmission Protocol |
| SOC | Security Operation Center |
| SLA | Service Level Agreement |
| SMS | Service Management System |
| SGNS | Skip Gram with Negative Sampling |
| SPSS | Statistical Package for the Social Sciences |
| TASA | Telecommunication Alarm Sequence Analyzer |
| TW | TimeWeaver |
| UMTS | Universal Mobile Telecommunications Service |
| UNMS | Unified Network Management System |

# 1. Introduction

Telecommunication networks are rapidly expanding in size and complexity, and managing them is getting increasingly complicated. The effort of identifying and fixing faults in telecommunication networks is a significant component of network management; failures that disrupt the network's services are costly to the operator. The quality of services also plays an important role in the growing competition between operators [3].

As heterogeneous networks become a reality through the deployment of micro, femto, and pico-cells, the complexity of O&M tasks increases [4]. Current O&M procedures must be extended to deliver efficient and high-quality communication services to end users in order to sustain such complex networks. Ethio Telecom (ET), which is the major internet and telephone service provider, has a total of 56.2 million customers all over the country. ET managed to implement numerous expansions in the network infrastructure to improve capacity as well as QoS for its customers.

The ongoing telecom *liberalization* process in ET will soon bring competition with other network operators. In a market where network and service offerings are nearly similar, how operators manage their networks to promote satisfaction is a differentiating feature. Fault Management's role is to limit the impact of network failures, which is a crucial goal in this procedure. FM comprises of problem recognition, problem notification, problem diagnosis, launch of corrective actions and restoration of initial settings after fixing the problem. Some of these tasks are labor-intensive [6]. FM begins by collecting real-time performance statistics from network devices and links, and then warns of failures and/or service degradation through various alarm sequences. When a problem is discovered, network engineers work to correct it. In ET, this process has been performed by a group of experts, located in the National Network Operation Center (NNOC), who use their experience and intuition to troubleshoot, localize, and solve faults by checking all the alarms collected in the different network segments. As a result, FM is currently the most time- and expertise-intensive of all network management processes [6].

A Unified Network Management System (UNMS), Resource Management System (RMS), and Service Management System (SMS) are used by NNOC to monitor network operations. ET uses these technologies for a variety of functions, including unified network administration, visual

fault detection and localization, real-time performance monitoring, and service deployment and monitoring. NNOC additionally comprises a trouble ticketing management system, a centralized power and environment management system, a fraud management system, and a Security Operation Center (SOC) in addition to the subsystems described above.

The growing ET networks report thousands of alarms to network management applications daily. An alarm is generated by a network element or its component to report an abnormal situation it has detected [5]. The flow of alarms offers a lot of comprehensive yet fragmented information concerning network faults. Fault management, as a network management application, examines the flow of alarms in order to isolate faults. Alarms are typically remote-implication offences, and they can only be examined effectively in the context of other alarms and other knowledge, making analysis complex. In addition, networks are large and alarms are very diverse, and alarms often occur in dense bursts [10]. To provide network management with self-healing and proactive fault management capabilities, significant amounts of data relating to faults, warnings, and KPIs must be monitored for a quick decision-making process. Network management systems are designed to be capable of managing network configurations, traffic routing, fault and performance management in real time [5].

Artificial intelligence (AI) techniques have proven useful in building network operation systems [l]. The proactive maintenance of networks is one area where such solutions can have a big influence. By recognizing potentially serious problems before they degrade, proactive network maintenance can greatly improve the quality and dependability of a network. This can be done by tracking network performance over time and identifying patterns in problems. Alarm prediction in the telecom domain is an extremely significant field of research due to our current dependency on telecommunications in our personal and professional lives. It has the potential to save businesses a great deal of money; personal users a large amount of inconvenience; and, in crisis situations, may save lives [8]. Telecommunication network performance management, on the other hand, is a difficult task since these networks often involve hundreds of components, and a problem with one component can quickly spread throughout the network. Identifying and isolating faults in these networks is one important aspect of managing network performance. In fact, in order to maintain the availability of these networks, it is critically important to predict a fault before it results in the total failure of a hardware component [11].

## 1.1 Statement of the Problem

Due to the increased complexity in traffic patterns, usage of networks and applications in the telecommunications network and services, as well as the presence of multiple equipment vendors and software systems, the need for predictive maintenance has been raised in order to decrease equipment breakdown [1]. Predictive maintenance is a set of procedures used by a company to limit the impact damage of a component in a telecom site in order to reduce operational costs and increase customer satisfaction.

| Months | Power | Transmition | OFC cut | IP | Total incidents per month | Total affected BTSs |
|---|---|---|---|---|---|---|
| **July** | **83** | **18** | **91** | **4** | 196 | 2077 |
| **August** | **53** | **23** | **91** | **4** | 171 | 1522 |
| **September** | **59** | **18** | **83** | **9** | 169 | 1650 |
| **October** | **66** | **10** | **89** | **7** | 172 | 1541 |
| **November** | **38** | **7** | **113** | **3** | 161 | 1094 |
| **December** | **47** | **10** | **85** | **10** | 152 | 2197 |

*Table 1.1 Ethio telecom 6-month Incident report, 2020.* *Source: Adapted from [2]*

Predictive maintenance implementation is a much more challenging research area. The different kinds and formats of data sources make preprocessing of the data a tough process. In this area, limited activities have been performed so far. In this respect, this research will provide additional insight and provide a different alarm prediction methodology for proactive fault management.

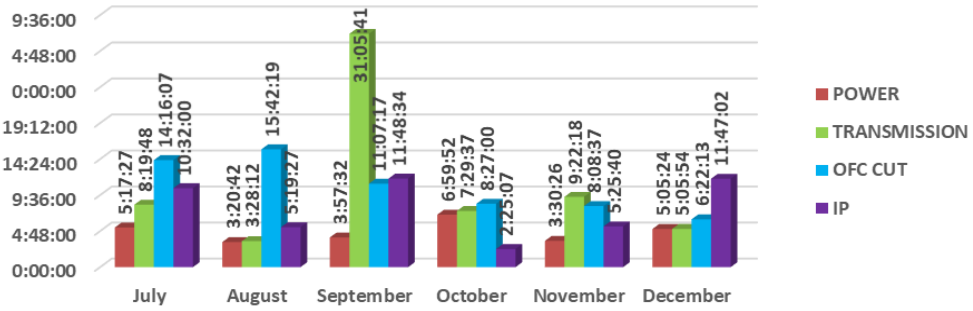


*Figure 1.1 Six month total average restoration time per domain, 2020.* *Source: Adapted from [2]*

Table 1.1 shows ET's six-month incident report. From the report, we can see that many incidents are registered every month due to power, IP, OFC cut, and transmission problems with several affected BTSs. Figure 1.1 also shows the average restoration time for these incidents. From the figure, we can observe that a minimum of 3 and a maximum of 31 hours of incident restoration time is needed to restore the incidents. As the amount of restoration time became longer, this resulted in a revenue loss as well as poor QoS.

To minimize the restoration time and prevent a network failure, it is important to follow proactive maintenance approaches which involve predicting alarms before a fault results in a total hardware failure. The approach gives a performance improvement on the network as well as decreasing revenue loss. Last but not least, it improves customer satisfaction.

## 1.2 Objective

### 1.2.1 General Objective

The overall objective of this research is to predict alarms in fault management with the help of deep learning algorithms, which are LSTM, Bi-LSTM, and GRU. For the success of this research, five-month historical alarm data has been collected from the Element Management System (EMS) of ET.

### 1.2.2 Specific Objectives

The specific objectives of this research work are:

- Study the methodology used in alarm prediction in order to select a suitable approach for this work;
- A literature review of different works related to alarm prediction, pattern discovery, deep learning, word embedding, association rules, frequent pattern algorithms, and fault management;
- Grasp the three deep learning approaches (LSTM, Bi-LSTM, GRU) that are widely used for NLP purposes and suitable for alarm prediction;
- Collecting data, preprocessing and converting it to binary transaction matrix form for discovering patterns;

- Perform association rule mining using Rapidminer studio, a knowledge base on discovered rules.
- Feed those rules to train and evaluate the performance of LSTM, Bi-LSTM and GRU on different feature representation techniques.

## 1.3 Methodology

To accomplish this research, the methodology followed is shown in Figure 1.2. It starts from reviewing different literature to get an idea for thesis writing, then focusing on alarm prediction-related works, especially in the telecom domain, is a good way to start. From the knowledge grasped, a suitable alarm prediction approach is selected and data collection is performed from the EMS of ET. Data preprocessing, one hot representation of alarms, association rules mining are data analysis steps following the first steps. The knowledge discovered from the data mining process is an input to the deep learning algorithms (LSTM, Bi-LSTM, and GRU). The three deep learning algorithms are trained, tested, and evaluated. Then, based on the model performance, tuning the hyper parameters on model training is conducted.

Different programming languages and packages are used to analyze data, train and visualize the results.

- Python
- R
- Rapidminer with weka extension

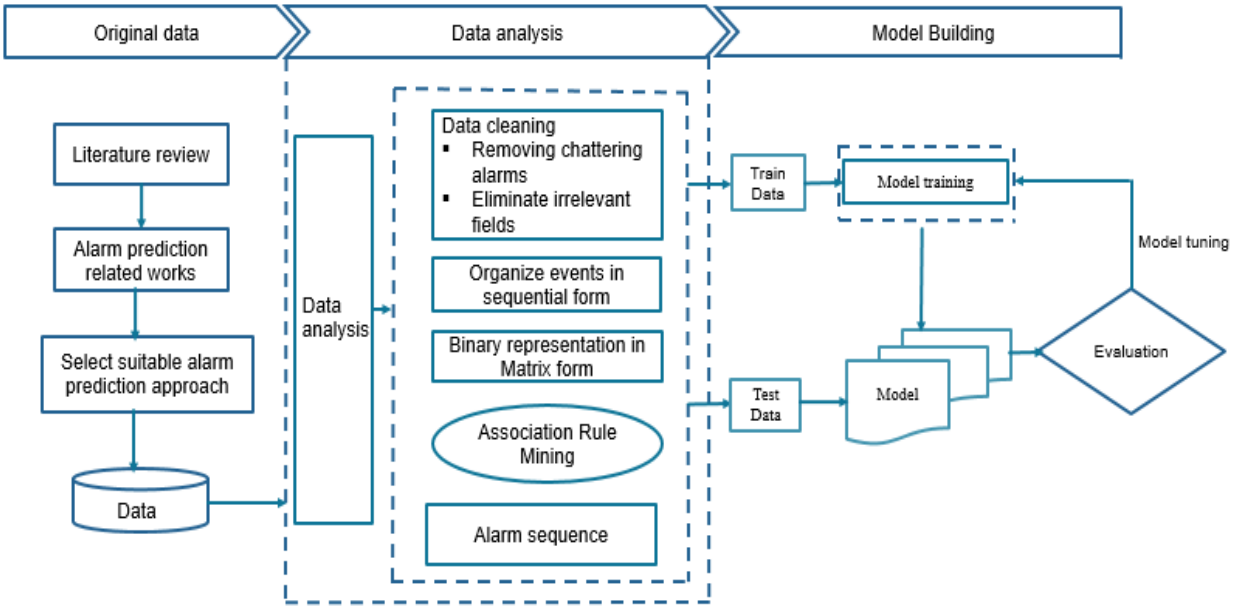


*Figure 1.2 Research methodology flow chart.*

## 1.4 Literature Review

Telecommunications companies were among the first to adopt data mining and data stream mining for applications like marketing and fraud detection, in addition to fault prediction. The following is a collection of research involved in fault prediction in telecommunication networks.

Zargarian used unsupervised machine learning for mining alarm logs and presented methods for semi-automatic discovery of patterns in databases [7]. Different alarm datasets are collected and employed frequent pattern mining to analyze data characteristics, dependency, correlation and anomalies. The pattern discovery follows two methodologies. The first is between separated network devices, and the second considers the device type. The alarm data is preprocessed and changed to binary matrix format for the frequent pattern mining algorithm. Rules are extracted and visualized to find the most important rules. The significance of the rules is measured with their lift, confidence, and support. From the comparison between the two methodologies, the second approach extracts more general association rules as it considers type rather than the separated device.

Shuang Cai, Ahmet Palazoglu, Laibin Zhang, Jinqiu Hu this paper presents an alarm prediction method based on word embedding and recurrent neural networks to predict the next alarm in a

process setting [6]. Their research is inspired by natural language processing in performing alarm prediction. Industrial alarms are segmented in sequence with a time threshold of 30 minutes, and they consider that alarms within this period have a relationship with one another. The SGNS word embedding method is used for mapping alarms into vectors, then the LSTM model is employed. In addition, the model is compared with the N-gram model and the prediction accuracy of the proposed approach outperforms it. The shortcoming of this paper is that it does not use knowledge discovery methods to extract the hidden pattern between alarms. It only assumes that there exists a relationship within the sequence at the given threshold.

Most of the alarm prediction approaches depend on association rule mining and apriori for prediction purposes. Wrench, C., Stahl, F., Le, T., Di Fatta, G., Karthikeyan, V., and Nauck introduce a method of rule induction for predicting and describing target alarms in telecommunication networks [8]. They implement ITRULE with modification as a rule induction algorithm and use 10,000 and 20,000 instances as training and test sets. The rule induction produces human-readable rules on the IP network and classifies instances. If an instance fulfills the criteria in the left hand side, then it is most likely it belongs to the class on the right hand side.

I. Caravela, A. Arsenio, and N. Borges developed a closed-loop automatic data mining approach for preventive network monitoring [9]. They use an apriori algorithm to discover sequential patterns from historical alarm data and build a knowledge base. Training the pattern from the knowledge base is implemented using an Acceptor Finite State Machine (AFSM). The prediction accuracy scores correct predictions of 72 % related to a 28 % rate of false positives. Although the patterns from the knowledge base are 90% accurate,

Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivone describe the Telecommunication Alarm Sequence Analyzer (TASA) [10]. It is a novel system for discovering knowledge and correlation between telecommunication alarm databases and episode rules. The episode rules show patterns found in the alarm dataset in human language form. The system supports two processes of knowledge discovery: pattern discovery and pattern presentation. Investigation of rules is left to the experts as the system only discovers all rules that satisfy the specified criteria. The system is developed by telecommunication companies and it is in prototype use.

With Gary M. Weiss, the paper describes the prediction of telecommunication equipment failures from sequences of network alarms [11]. For finding telecommunication equipment faults from network alarm logs, TimeWeaver (TW), which is a temporal data mining system, is used. A total of 148,886 4ESS switch alarms were collected for two weeks and used as a target dataset. They consider the (time the alarm was generated, a unique identifier for the device, the type of device, diagnostic code and severity) as a variable from the dataset. The dataset is split into 70% training and 30% for testing, so as TW is applied to the training set. With a monitoring time of eight hours and varying warning times, the results show that better prediction is achieved as the warning time becomes shorter. It shows that alarms occurring near the failure are important in predicting the failure.

A different method for alarm prediction is conducted in some papers, and this paper is one of them by I. Khan, Joshua Z. Huang, and N. Thanh Tung, learning time-based rules for prediction of alarms from telecom alarm data using Ant Colony Optimization (ACO) [12]. They recommend that for a large transaction, the sequential data mining process is not scalable, so for extracting time-based rules from the large dataset they implement an ant colony optimization process. Two real time datasets from Nokia Simmons (NSN) and Ericsson are used for the experiment. In the dataset, five attributes (start time, alarm severity, alarm origin, alarm code, alarm type) and six classes are selected. The extracted time-based rules from the process indicate the association between the alarm start time and alarm type. Minimum support and confidence are the criteria for rule selection, and finally, the prediction accuracy of the proposed method is compared with the TimeSeluth system. With 91.8% accuracy, the TimeSeluth and 94.2% of the results are scored by the ACO based algorithm.

## 1.5 Scope and Limitation

The scope of this research is to predict alarms in fault management, and it is limited to BTS alarms only, as there are many sub-systems generating different types of alarms within the fault management system. In the future, the scope can be widened to include other types of alarms from other devices. The second limitation is on the data collection process. The EMS can only store a 5-month history of alarms because of storage problems, so the alarm log mining process is restricted to this period only.

## 1.6 Contribution

The contribution of this thesis is as follows:

- It motivates the company to change its approach from reactive maintenance or reactive monitoring in fault management to a proactive maintenance approach, which has its own advantages in terms of preserving equipment, keeping it in the best possible condition and making unexpected failures happen rarely.
- Obtained rules can aid domain experts in recalling alarm correlation patterns as they show current network status.
- The comparison of the deep learning algorithms which are LSTM, Bi-LSTM, and GRU will give direction for students interested in studying in the area of prediction.
- Limited research is conducted in telecom alarm prediction; therefore, it can act as an additional reference to the scientific community.

## 1.7 Thesis Organization

The rest of the thesis is organized as follows: Chapter two discusses the technical background of fault management, data mining, deep learning, and discusses BTS alarms. Chapter three shows the alarm prediction methodology starting from the process of knowledge discovery in historical alarm data and explains the steps from data collection to rule generation. Chapter four discusses model training and evaluation from parameter setup to three deep learning models. LSTM, GRU, Bi-LSTM are trained using domain trained word embedding and pretrained word embedding. Chapter five results and discussions are presented for the LSTM, Bi-LSTM and the GRU based models. Chapter six presents the conclusion drawn from the research and recommendation for future works.

# 2. Technical Background

## 2.1 Fault Management in Telecom Networks

A fault is a disorder occurring in the hardware or software of the managed network. Faults happen within the managed network or its components [13], and fault management (FM) is the process of locating, analyzing, fixing, and reporting network problems such as link failures and network overload [14].

Fault management is an important but difficult area of telecommunication network management. Networks produce large amounts of alarm information which must be analyzed and interpreted before faults can be located [15]. The Fault management system (FMS) involves management of faults within an organization. It will assist engineers in the network operations center to log faults and be able to escalate faults to expected field engineers [27]. The field engineers are also expected to report observations and activities carried out with respect to a particular fault to ensure the fault is closed appropriately from the system [21].

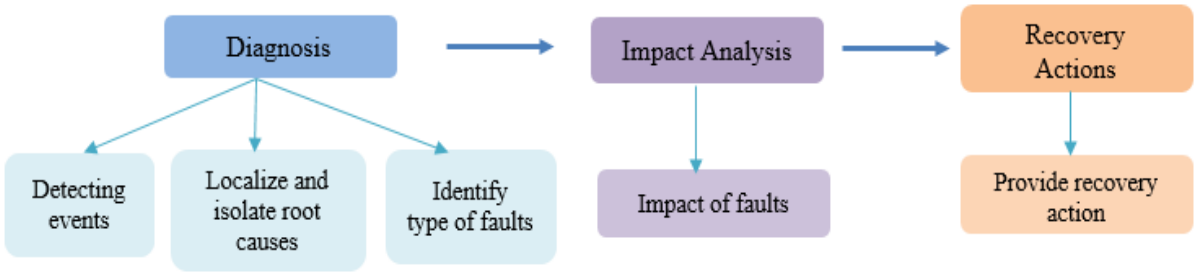


*Figure 2.1 The Fault Management Process.* *Source: Adapted from [14].*

Figure 2.1, shows the FM steps from fault detection to healing. Fault diagnosis aims at achieving three complementary tasks: fault detection, localization and identification [14]. In the following subsections, the different steps of fault management are discussed.

### 2.1.1 Fault Detection

This is the first stage of a fault management system. Fault detection in sensor networks depends on the type of failures that occur on a network [21]. This stage determines whether the system is

operating normally or if there is a problem. When an error causes network devices or software to malfunction, resulting in symptoms or alarms, this is referred to be a failure. Any network service provider's Service Level Agreement (SLA) and performance requirements may have an impact on normal and aberrant behavior. The behavior and causes of many types of faults can be classified. Faults can be permanent, transient for a short period, or repetitive. They can originate from maintenance, configuration errors, or malicious intrusions [14].

### 2.1.2 Impact Analysis

Impact analysis or fault Diagnosis is the process of properly identifying the cause of a problem or the cause of an alarm. It is the process of properly troubleshooting the failure or fault in the system to determine the root cause of the failure [21]. Impact analysis is concerned with identifying the specific nature of faults (e.g. their size, propagation, importance, damage etc.). Moreover, the impact analysis helps administrators understand the risks of potential process or product failures. Several methodologies have been developed to quantify the effects and impacts of failures: Failure Modes and Effects Analysis (FMEA); Failure Modes, Effects, and Criticality Analysis (FMECA) are all graph-based [14].

### 2.1.3 Recovery Action

This is the stage where the network is restructured or reconfigured such that failures do not affect or impact the network performance any further [21]. In conventional networks, most recovery activities were carried out by network professionals because most failures were caused by hardware server crashes, which required administrators to repair or replace the equipment. In the network link recovery, however, some auto-recovery steps were supported. The network path recovery process handles two general failure recovery approaches: protection and restoration [30]. In the protection approach, the routes are reserved and computed before a failure occurs. On the other hand, restoration, the new link allocation occurs after the failures. Restoration may increase the recovery time since it is preceded by protection, which may cause congestion due to link reservation [14].

## 2.2 Alarms

An alarm is an indication of a probable malfunction that can be seen. Devices or network management systems can create alarms to provide information about potential defects that could

cause errors. Alarms are critical because they enable the detection of faults and the implementation of corrective action. Many alarms may appear in a network because a single fault may cause multiple alarms and because some alarms may be triggered in situations where no fault has occurred [17].

Alarms are sent to management centers when faults in a telecommunication network occur. An alert is a communication that a network element sends out when it detects a problem. Unfortunately, because a network node has such a limited view of the network, it can only describe the symptoms of the issue from that perspective. On the other hand, one fault can result in a number of different alarms from several network elements [15].

| Hardware fault | Related alarms |
|---|---|
| Board overload alarm | Board Overload |
| Alarms related to RF modules | RF Unit VSWR Threshold Crossed |
| | RF Unit RX Channel RTWP/RSSI Unbalanced |
| Cell capability degraded alarm | Cell Capability Degraded |
| Alarms related to CPRI links | RF Unit Maintenance Link Failure |
| | BBU CPRI Interface Error |
| | BBU CPRI Optical Interface Performance Degraded |
| | RF Unit Optical Interface Performance Degraded |
| Alarms related to clock sources | IP Clock Link Failure |
| | System Clock Unlocked |
| | RF Unit Clock Problem |
| | System Clock Failure |
| | Base Station Frame Number Synchronization Error |
| Alarms related to transmission faults | IP Path Fault |
| | SCTP Link Fault |
| | User Plane Path Fault |
| | S1 Interface Fault |

*Table 2.1 Example of BTS Hardware faults.* *Source: From [18].*

## 2.3 Data Mining

Data mining refers to extracting or mining knowledge from large amounts of data. The term is actually a misnomer. As a result, data mining should have been renamed knowledge mining, with an emphasis on mining from large amounts of data [45]. It is a computational process that involves approaches from artificial intelligence, machine learning, statistics, and database systems to uncover patterns in massive data sets. The overall purpose of the data mining process is to extract information from a data set and transform it into a structure that can be used.

Data mining is a logical process that is used to search through large amounts of data in order to find useful data. The goal of this technique is to find patterns that were previously unknown. Once these patterns are found, they can further be used to make certain decisions for the development of their businesses [45].

Three steps involved are

- Exploration
- Pattern identification
- Deployment

Data mining itself is generally split into two categories: descriptive data mining and predictive data mining. Descriptive data mining explores interesting patterns to describe the data, while predictive data mining forecasts the behavior of the model based on the available data set [19].

**Predictive:** Prediction is a type of data mining that focuses on finding relationships between independent variables and between dependent and independent variables. Predictive data mining can be used to forecast explicit values using data patterns. Predictive Data Mining is usually applied with the goal of identifying a statistical or neural network model that can be used to predict some kind of interesting result. Prediction analysis techniques can be used, for instance, in sales to predict future profit based on previous sales activity [19].

**Descriptive:** A descriptive model identifies patterns or relationships in data. Unlike the predictive model, a descriptive model serves as a way to explore the properties of the data

examined, not to predict new properties. Clustering, summarization, association rules, and sequence discovery are usually viewed as descriptive in nature [20].

### 2.3.1 Market Basket Analysis

Market basket analysis is a method of identifying object associations that "go together" in a commercial context. Market basket analysis, in actuality, goes beyond the supermarket scenario from which it gets its name. Market basket analysis is the analysis of any collection of items to identify affinities that can be exploited in some manner [16].

| TID | Items |
|---|---|
| 1 | {Bread, Milk} |
| 2 | {Bread, Diapers, Beer, Eggs} |
| 3 | {Milk, Diapers, Beer, Cola} |
| 4 | {Bread, Milk, Diapers, Beer} |
| 5 | {Bread, Milk, Diapers, Cola} |

*Table 2.2. An example of market basket transactions Source: Adapted from [23]*

It monitors purchase trends and improves customer service by focusing on client baskets. In retail enterprises, it's an important part of analytical CRM. A pattern can be discovered by examining reoccurring patterns in order to offer comparable goods together, and therefore sales can be increased. Sales on many levels of goods categorization and client segments can be easily tracked. For example, if the company knows which items are often purchased together, they can create new offers on those products in order to increase the sales of those items, or they can create combo promotions which increase the sales of their products [22].

For example, the following rule can be extracted from the data set shown in Table 2.2:

{Diapers} ~ {Beer}

The rule suggests that a strong relationship exists between the sale of diapers and beer because many customers who buy diapers also buy beer. Retailers can use this type of rules to help them identify new opportunities for cross selling their products to the customers [23].

| TID | Bread | Milk | Diapers | Beer | Eggs | Cola |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 5 | 1 | 1 | 1 | 0 | 0 | 1 |

*Table 2.3 A binary 0/1 representation of market basket data* *Source: Adapted from [23]*

Let *I = i1, i2,,, id* be the set of all items in a market basket and *T = t1, t2,..., tN* be the set of all transactions. Each transaction *ti* contains a subset of items chosen from *I*. In association analysis, a collection of zero or more items is termed an itemset. If an itemset contains *k* items, it is called a k-itemset. For instance, "Beer, diapers, milk" is an example of a 3-itemset. The null (or empty) set is an itemset that does not contain any items. A transaction *tj* is said to contain an itemset X if X is a subset of *tj* [23]. For example, the second transaction shown in table 2.2 contains the itemset "Bread, Diapers" but not "Bread, Milk". An important property of an itemset is its support count, which refers to the number of transactions that contain a particular itemset. Mathematically, the support count, (X), for an itemset X can be stated as follows in Equation (2.1) [24].

$$\sigma(X) = |\{t_i \mid X \subseteq t_i, t_i \in T\}| \quad (2.1)$$

Market basket data can be represented in a binary format as shown in Table 2.3 where each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise. Because the presence of an item in a transaction is often considered more important than its absence, an item is an asymmetric binary variable [23].

## 2.4 Association Rules

The data mining technique most closely identified with market basket analysis is association analysis, which automatically generates association rules. Association analysis is a type of undirected data mining that finds patterns in the data where the target is not specified beforehand.

Whether the patterns make sense is left to human interpretation [25]. Association rule mining is seen as a two-step approach:

**1. Frequent Itemset Generation:** Find all frequently occurring itemsets with a low support count using the Frequent Itemset Generation method. Usually, interesting relationships and correlations between itemsets in transactional and relational databases are discovered through frequent mining. Frequent Mining shows which things appear together in a transaction or relationship. The discovery of frequent itemsets is accomplished in several iterations. Counting new candidate itemsets from existing itemsets requires scanning the entire training data. In short, it involves only two important steps [22]:

a. Pruning

b. Joining

2. **Rule Generation:** Generate a list of all the association rules from frequently used itemsets. Calculate Support and Confidence for all the rules. Prune rules which fail minimum support and minimum confidence thresholds [22]. There are three indexes which are commonly used to understand the presence, nature and strength of an association rule.

**Support:** It is the measure of how often the collections of items in an association occur together as a percentage of all transactions. Support (s) for an association rule X=>Y is the percentage of transactions in the database that contain X U Y shown in Equation (2.2) [27]. Every association rule has support. The rule that has very low support may occur simply by chance [26].

$$\text{Support, } s(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{2.2}$$

**Confidence:** Confidence for an association rule X=>Y is the ratio of the number of transactions that contain both antecedent and consequent to the number of transactions that contain only antecedent. A rule with low confidence is not meaningful. Confidence () for an association rule X=>Y is the ratio of the number of transactions that contain X U Y to the number of transactions that contain X stated in Equation (2.3) [27] [26].

$$\text{Confidence , } c(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{2.3}$$

**Lift:** One parameter that is also used to determine important rules in association rules is lift ratio. Lift Ratio is a parameter used to determine whether or not the rules gained from association rules are strong. The Lift Ratio measures the possibility of X and Y occurring together divided by the possibility of X and Y occurring if they are independent events. The formula for the ratio is shown in Equation (2.4) [27].

$$\text{Lift Ratio} = \frac{\text{Support } (X \cap Y)}{\text{Support } (X).\ \text{Support } (Y)} \tag{2.4}$$

The strength of an association rule is measured by confidence and support. Because the transactional database is so vast, there's a greater chance of receiving too many unimportant and irrelevant rules. To avoid such errors, we often set a support and confidence level prior to the analysis, ensuring that only meaningful and interesting rules are generated in the end.

If lift is greater than 1, it suggests that the presence of the items on the LHS has increased the probability that the items on the RHS will occur on this transaction. If the lift is below 1, it suggests that the presence of the items on the LHS makes the probability that the items on the RHS will be part of the transaction lower [22]. If the lift is 1, it suggests that the presence of items on the LHS and RHS are independent. Knowing that the items on the LHS are present makes no difference to the probability that items will occur on the RHS. While performing market basket analysis, we look for rules with a lift of more than one [22].

## 2.5 Frequent pattern Mining algorithms

When mining with low minimum support values, a trustworthy frequent pattern mining method should give acceptable performance metrics such as low execution time and low memory use, and it should be scalable. The two most prevalent algorithms will be discussed in the subsections below.

- Apriori-based
- Pattern-growth

### 2.5.1 Apriori

Apriori-based algorithms follow the apriori property which states that "all nonempty subsets of a

frequent itemset must also be frequent." To be "frequent", a set must have support that is greater than the specified minimum support [19].

A k-itemset is frequent only if all of its sub-itemsets are frequent. This implies that frequent itemsets can be mined by first scanning the database to find the frequent 1-itemsets, then using the frequent 1-itemsets to generate candidate frequent 2-itemsets, and checking against the database to obtain the frequent 2-itemsets. This process iterates until no more frequent k-itemsets can be generated for some k. This is the essence of the Apriori algorithm [28].

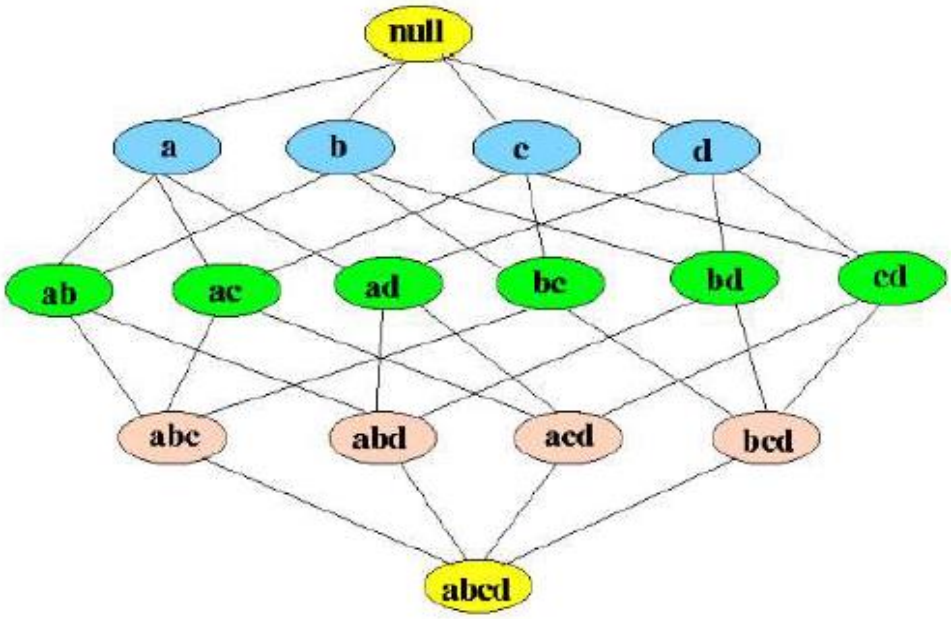


*Figure 2.2 An itemset lattice* *Source: Adapted from [28]*

To illustrate the idea, a lattice structure is used to enumerate the list of all possible itemsets. As you can observe in 2.2, the graph shows an itemset lattice for I= "a, b, c, d". Generally, a data set that contains k items can potentially generate up to $2^k - 1$ frequent itemsets without the null set. Because k can be very large in many practical applications, the search space of itemsets that is required to be explored is exponentially large [7].

### 2.5.2 Frequent pattern (FP) Growth

In the field of data mining, the most popular algorithm used for pattern discovery is the FP Growth algorithm. To deal with the two main drawbacks of the Apriori algorithm, a novel,

compressed data structure named the FP tree is constructed, which is a prefix-tree structure storing quantifiable information about frequent patterns. Based on the FP tree, a frequent pattern growth algorithm was developed [29].

It's a two-step approach. In the first step, a frequent pattern tree is constructed by scanning the database twice. In the first pass of the database, data is scanned and the support count for each item is calculated. Infrequent patterns are deleted from the list, and the remaining patterns are sorted in descending order. In the 2nd pass of the database, the FP Tree is built. In the 2nd step, using the FP growth algorithm, frequent patterns are extracted from the FP Tree [29].

FP-growth works in a divide and conquer way. The first scan of the database derives a list of frequent items in which items are ordered by frequency in descending order. According to the frequency descending list, the database is compressed into a frequent pattern tree, or FP-tree, which retains the itemset association information [28]. The FP-tree is mined by starting from each frequent length-1pattern (as an initial suffix pattern), constructing its conditional pattern base (a "sub database", which consists of the set of prefix paths in the FP-tree co-occurring with the suffix pattern), then constructing its conditional FP-tree, and performing mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated by a conditional FP-tree. The FP-growth algorithm transforms the problem of finding long, frequent patterns into searching for shorter ones recursively and then concatenating the suffix. It uses the least frequent items as a suffix, offering good selectivity. Performance studies demonstrate that the method substantially reduces search time [28].

## 2.6 Pattern Mining Tools

There is a large amount of data generated every second, and it is necessary to have knowledge of different tools that can be utilized to handle this large data and apply interesting data mining algorithms and visualizations in quick time [30].

There are numerous data mining tools available, including MonkeyLearn, Rapidminer, Oracle Data Mining, IBM SPSS Modeler, Weka, and others. Rapidminer is utilized to execute data mining tasks in this work. Rapidminer is a free, open-source data science platform with hundreds of algorithms for data prep, machine learning, deep learning, text mining, and predictive

analytics. It is one of the most effective predictive analytic methods available. It offers a comprehensive environment for deep learning. The tool can be utilized in a wide range of situations. It covers corporate applications, commercial applications, training, and education, among other things. Rapidminer offers the server both on-premises and in public and private cloud infrastructures. It has a client/server model as its base. Rapidminer comes with template based frameworks. Also, it enables speedy delivery with a reduced number of errors [30].

From Figure 2.3, non-programmers may design predictive processes for specific use cases like fraud detection and customer churn using a drag-and-drop interface and pre-built models. Meanwhile, programmers may personalize their data mining using Rapidminer's R and Python extensions. Rapidminer studio allows to visualize results after you've developed your workflows and analyzed your data to help you spot patterns, outliers, and trends in your data.

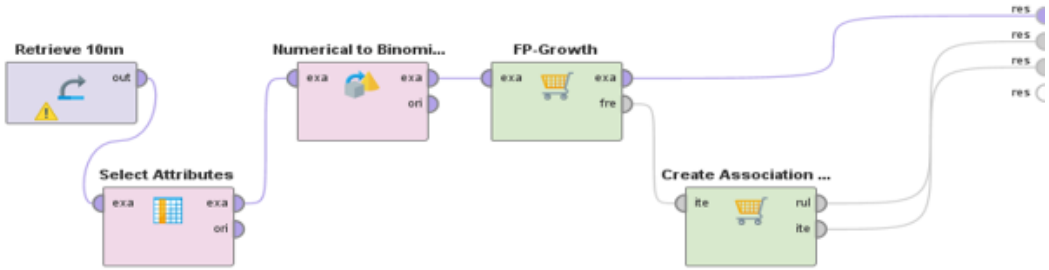


*Figure 2.3 Creation of Association rule in Rapidminer*

## 2.7 Deep Learning

Deep Learning is currently demonstrating near-state-of-the-art performance on a variety of tasks in a variety of domains, including speech recognition, computer vision, and natural language processing. There are a huge number of different variants of deep architectures, such as Deep Neural Network (DNN), Deep Belief Network (DBN), Deep Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) [36].

Deep learning algorithms have lately demonstrated outstanding performance in Natural Language Processing applications such as sentiment analysis and emotion recognition across a variety of datasets. Such models do not need any pre-defined hand-picked features, but they learn sophisticated features from the input datasets by themselves. In such a domain, word embedding

has been widely used as a way of representing words in sentiment analysis tasks, proving to be very effective [32].

Word embeddings are one of the most important aspects of representing textual data and feeding Machine Learning algorithms. They are word representations that have been mapped to real-number vectors. Neural Networks are used in the first word embedding model (Word2Vec). Word2vec is a popular sequence embedding method that transforms natural-language into distributed vector representations. It can capture contextual word-to-word relationships in a multidimensional space and has been widely used as a preliminary step for predictive models in semantic and information retrieval tasks [33].

Almost every Natural Language Processing (NLP) model used in practice today uses word embeddings. The reason for such a massive adoption is their effectiveness. By translating a word to an embedding, it becomes possible to model the semantic importance of a word in a numeric form and, thus, perform mathematical operations on it [32]. In the following subsections, the types of algorithms used in deep learning are discussed.

### 2.7.1 Simple Recurrent Neural Network (Simple RNN)

A simple RNN is essentially a collection of common neural networks arranged together, each of them transmitting a message to another. In other words, these networks have a memory that stores knowledge about the data seen, but their memory is short-term and cannot maintain long-term time series [46]. Figure 2.4 displays a simple RNN. A simple recurrent network has only one internal memory $h_t$ which is computed from Equation (2.5) [35].

$$h_t = g\left(W x_t + U_f h_{t-1} + b\right) \tag{2.5}$$

where $g$ () denotes an activation function, $U$ and $W$ are flexible weight matrices of the h layer, $b$ is a bias, and X is an input vector [35].

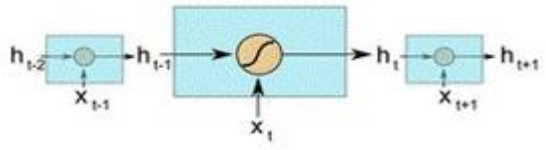


*Figure 2.4 Simple RNN* *Source: Adapted from [35]*

### 2.7.2 Long Short-Term Memory (LSTM)

LSTM is an implementation of the Recurrent Neural Network. LSTM partly addresses a major limitation of RNN, i.e., the problem of vanishing gradients, by letting gradients pass unaltered [34]. LSTM is a kind of model or structure for sequential data developed for the advancement of RNN. It uses a special combination of hidden units, elementwise products, and sums between units to implement gates that control "memory cells." These cells are intended to retain information without modification for extended periods of time [34]. The ability of LSTM to learn long-term dependency, which is not feasible with simple RNNs, is its most valuable characteristic. The weight values on the network must be updated to forecast the future step, which necessitates the preservation of data from the previous steps. A simple RNN can learn just a small number of short-term associations and cannot learn long-term series. However, LSTM can learn these long-term dependencies properly, and LSTM has three gates: input, forget, and output (Figure 2.5). The forget gate is embedded to indicate how much the previous memory remembers and how much it has forgotten. For LSTM, the hidden state $h_t$ is computed as follows in Equations (2.6), (2.7) (2.8) (2.9) (2.10) and (2.11) [35]:

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i) \tag{2.6}$$

$$f_t = \sigma\left(W_f X_t + U_f h_{t-1} + b_f\right) \tag{2.7}$$

$$O_t = \sigma(W_o X_t + U_o h_{t-1} + b_o) \tag{2.8}$$

$$\tilde{C}_t = \tanh(W_c X_t + U_c h_{t-1} + b_c) \tag{2.9}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.10}$$

$$h_t = \tanh(C_t) * O_t \tag{2.11}$$

where $\dot{i}t$, *ft*, and *Ot* are the input, forget, and output gates at time *t*, respectively; *Wi*, *Wf*, *Wo*, and *Wc* are weights that map the hidden layer input to the three gates of input, forget, and output while *Ui*, *Uf*, *Uo*, and *Uc* weights matrices map the hidden layer output to gates; *bi*, *bf*, *bo*, and *bc* are vectors. Moreover, *Ct* and *ht* are the outcome of the cell and the outcome of the layer, respectively [35].

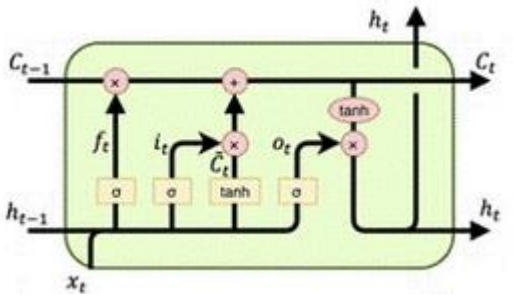


*Figure 2.5 Long Short-Term Memory* *Source: Adapted from [35]*

### 2.7.3 Gated Recurrent Unit (GRU)

GRU is a simple type of LSTM. The variance with LSTM is that GRU merges the input and forget gates and converts them with an update gate. Therefore, GRU has fewer parameters than LSTM which makes training easier. For GRU, the output value of $h_t$ *is* computed as follows in Equation (2.12), (2.13), (2.14) and (2.15) [31] [35]:

$$z_t = \sigma(W_z X_t + U_z h_{t-1} + b_z) \quad (2.12)$$

$$r_t = \sigma(W_r X_t + U_r h_{t-1} + b_r) \quad (2.13)$$

$$\widetilde{h_t} = \tanh(W_h X_t + (r_t * h_{t-1}) U_h) \quad (2.14)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \bar{h}_t \quad (2.15)$$

where *r* is a reset gate, and *z* denotes an update gate. The reset gate indicates how the new input can be combined with earlier memory. The update gate also indicates how much previous memory is kept. If the update gate is 1, the previous memory is fully preserved, and if it is 0, the previous memory is completely forgotten. There is a forget gate in the LSTM that automatically determines how much of the previous memory is maintained whereas, in GRU, all previous memories are maintained or completely forgotten (Figure 2.6). For some problems, GRUs can provide a performance comparable with LSTMs, but with a lower memory requirement [34,36] [35].

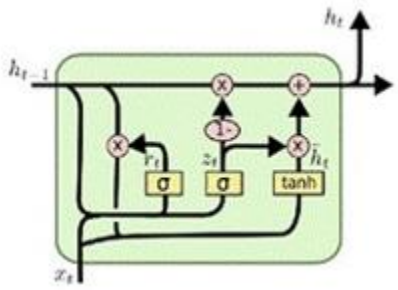


*Figure 2.6 Gated Recurrent Unit* *Source: Adapted from [35]*

The GRU is essentially a variant of vanilla LSTM with a forget gate. Since one gate is missing, the single GRU cell is less powerful than the original LSTM. The GRU cannot be taught to count or to solve context-free language and also does not work for translation [31].

### 2.7.4 Bidirectional LSTM (Bi-LSTM)

Bi-LSTMs are proven especially helpful in the occasions where the context of the input is needed. It is very useful for works like sentiment classification. In unidirectional LSTM information flows from backward to forward. On the contrary in Bi-directional LSTM information not only flows backward to forward but also forward to backward using two hidden states. Hence Bi-LSTMs understand the context better [37].

The structure of a bidirectional network is displayed in Figure 2.7. Given this figure, these networks have a structure with a forward and backward LSTM layer. The forward layer output order, $\rightarrow ht$, is repeatedly computed via inputs in a positive order from time $T-n$ to time $T-1$, while the backward layer outcome order, $\leftarrow ht$, is computed using the inverted inputs from time $T-n$ to $T-1$ [35]. The output of both forward and backward layers is computed in a similar manner to the unidirectional LSTM. In the Bi-LSTM layer, *Yt* is computed from Equation (2.16) [35]:

$$Y_t\left(\overrightarrow{h_t}, \overleftarrow{h_t}\right) \tag{2.16}$$

Where, σ is a function used to merge two h→ and h← outputs [35].

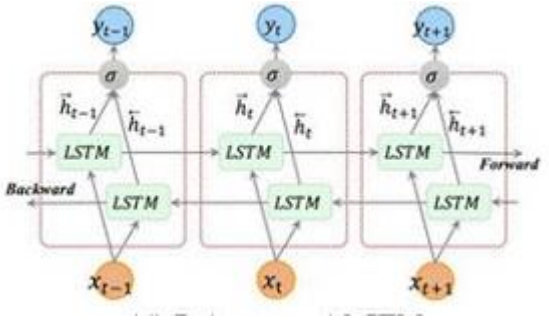


*Figure 2.7 Bidirectional LSTM* *Source: Adapted from [35]*

## 2.8 Optimization techniques

In the following discussions, several of the powerful strategies that may be applied to deep learning algorithms to reduce training time and maximize model performance are explored.

**Back propagation:** While solving an optimization problem using a gradient based method, backpropagation can be used to calculate the gradient of the function for each iteration [38].

**Stochastic Gradient Descent**: Using the convex function in gradient descent algorithms ensures finding an optimal minimum without getting trapped in a local minimum. Depending upon the values of the function and learning rate or step size, it may arrive at the optimum value in different paths and manners [47].

For each training sample, stochastic gradient descent calculates the gradient and updates the parameters. However, because of the high variance of different training samples, frequently updating parameters will cause the objective function to fluctuate a lot. Although a small step size can let SGD convergent to a good point, it makes the training slow. Moreover, when we use GPUs to conduct the computation, the frequent data commutation between GPU memory and local memory also decrease the efficiency [36].

**Learning Rate Decay:** The learning rate of stochastic gradient descent algorithms can be changed to improve performance and reduce training time. The most generally used strategy is to progressively reduce the learning rate, which allows us to make substantial modifications at first and then gradually reduce the learning rate throughout the training phase. This allows fine-tuning the weights in the later stages. An improper learning rate, whether too high or too low, results in a model with low effective capacity due to optimization failure [37].

**Dropout:** The overfitting problem in deep neural networks can be addressed using the drop out technique. This method is applied by randomly dropping units and their connections during training [48]. Dropout offers an effective regularization method to reduce overfitting and improve

generalization error. Dropout gives an improved performance on supervised learning tasks in computer vision, computational biology, document classification, speech recognition. Dropping units less often gives the units more opportunities to "conspire" with each other to fit the training set [37].

**Max-Pooling:** In max-pooling a filter is predefined, and this filter is then applied across the non-overlapping sub-regions of the input taking the max of the values contained in the window as the output. Dimensionality, as well as the computational cost to learn several parameters, can be reduced using maxpooling [49].

**Batch Normalization:** Batch normalization is one of the most exciting recent innovations in optimizing deep neural networks and it is actually not an optimization algorithm at all. Instead, it is a method of adaptive re parametrization, motivated by the difficulty of training very deep models [37].

Batch normalization reduces covariate shift, thereby accelerating deep neural network. It normalizes the inputs to a layer, for each mini-batch, when the weights are updated during the training. Normalization stabilizes learning and reduces the training epochs. The stability of a neural network can be increased by normalizing the output from the previous activation layer [38].

**Skip-gram**: Skip-gram can be used to model word embedding methods. When two vocabulary terms have a similar context in the skip-gram model, they are considered identical. For instance, the words "cats are mammals" and "dogs are mammals" are both coherent sentences that imply "are mammals." Skip-gram can be implemented by considering a context window containing n terms and train the neural network by skipping one of this term and then use the model to predict skipped term [50].

**Transfer learning:** In transfer learning, a model trained on a particular task is exploited on another related task. The knowledge obtained while solving a particular problem can be transferred to another network, which is to be trained on a related problem. This allows for rapid progress and enhanced performance while solving the second problem [38].

# 3. Alarm Prediction Methodology

This research follows the knowledge discovery process as a method to extract valuable information from alarm logs. The extracted valuable information or pattern from the process of knowledge discovery is used to train deep learning algorithms for the purpose of predicting BTS alarms. This chapter mainly focused on the Knowledge Discovery in Databases (KDD), a process followed in this research work.

The automatic, exploratory examination and modeling of huge data repositories is known as KDD [39]. The coordinated process of uncovering legitimate, unique, valuable, and intelligible patterns from huge and complicated data sets is known as knowledge discovery. Data Mining (DM) is the core of the KDD process, involving the inferring of algorithms that explore the data, develop the model and discover previously unknown patterns. The model is used for understanding phenomena from the data, analysis and prediction [39].

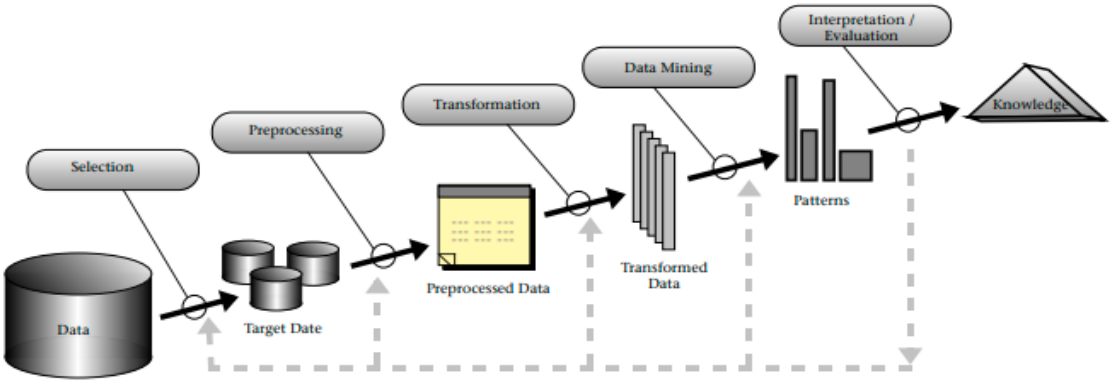


*Figure 3.1 KDD process* *Source: Adapted from [40]*

The KDD is a collaborative and iterative approach (with many decisions made by the user). The KDD process entails using the database, as well as any necessary selection, preprocessing, subsampling, and transformations; applying data-mining methods (algorithms) to enumerate patterns from it; and evaluating the data-mining products to determine which subset of the enumerated patterns is considered knowledge. The computational mechanisms by which patterns are extracted and enumerated from data are dealt with in the data-mining component of the KDD process. The overall KDD process (Figure 3.1) includes the evaluation and possible interpretation of the mined patterns to determine which patterns can be considered new knowledge [40]. The steps followed in KDD process is described in the next sections.

## 3.1 Data Collection and Preprocessing

Data pre-processing, also known as data preparation, focuses on two issues: first, the data must be structured in a format suitable for data mining methods, and second, the data sets used must provide the highest performance and quality for data mining models.

Data in the real world is frequently incomplete, noisy, and inconsistent. This might result in low-quality data collection and, as a result, low-quality models based on that data. The KDD technique, and therefore data mining processes are frequently used to combine data from numerous sources into a single collection. Although the KDD technique works with large data sets, and more data should theoretically lead to more precise findings, real-world experience has shown that this isn't always the case. Data dimensionality can be minimized by detecting and removing as much irrelevant and redundant content as possible, allowing for faster and more effective pattern generation.

To overcome these issues, data preprocessing includes actions such as data cleansing, data integration, data transformation, and data reduction. The primary goals of data reduction are to avoid over fitting and improve model performance, or prediction performance in the case of supervised classification; to provide faster and more cost-effective models; and to gain a deeper understanding of the underlying processes that generated the data [41].

In this thesis, historical alarm data from ET is used. The data is extracted from the element management system (EMS), which is located in the network operation center. The data collection includes a list of alarms generated in the period of five months between October 2020 and February 2021, with a total of 1,469,058 alarms.

### 3.1.1 Alarm Features

The data contains more than 15 features which describe the severity, network type, source of the alarm, including location, occurrence time, and other extra information. For reducing complexity only, the main features are considered to represent an event. Those features are described below.
**Severity:** The severity level indicates the severity and importance of an alarm. The categories can be Critical, Major, Minor or Warning. For a critical alarm, action must be taken immediately in

order to restore the service. In a major alarm, action must be taken as soon as possible because of service degradation. A minor alarm indicates a problem that does not yet affect service, but may lead to a problem if not handled properly. The warning alarm indicates a future problem that may affect the service.

**Alarm Name:** The alarm name tells the specific problem causing the fault on the device and there are 115 BTS alarm names present in the dataset.

**Alarm ID:** It's a unique number assigned for alarm names.

**NE type:** Specifies the technology (GSM, UMTS, LTE) and equipment type. It could be a base station controller, a base transceiver station, a radio network controller, or any number of systems the alarm belongs to.

**Alarm source:** The full device name generating the alarm, including its geographical location and vendor

**Occurrence time:** The first notice by the element management system

**Location Information:** Detailed information on the subrack number, IP address, slot number, board type, service type, and link number related to the event.

### 3.1.2 Statistics of collected Data

In the above section, features that will be used for representing alarms are explained. It is also necessary to see the general stastics of the data to show data characteristics.

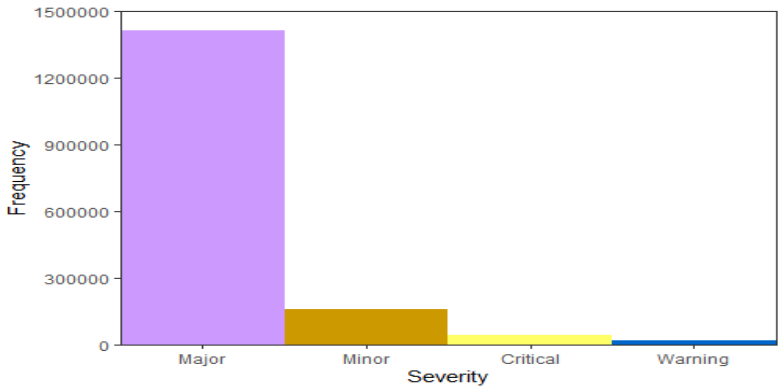


*Figure 3.2 Alarms categorized by severity*

Alarms categorized by severity Figure 3.2 shows that the largest share of data belongs to major alarms. As can be seen, there is a class imbalance in the dataset. Major and critical alarms together occupy 89% of the other alarms. Figure 3.3 also gives an indication that most of the

alarms are from communications alarm types. The alarm types are divided based on the property of the fault. The loss of communication between network devices is indicated by a communication alarm. Environmental alarms signal license or communication issues, whereas equipment alarms signal physical resource issues. Processing error alarms indicate software failures or faulty fault management techniques, while QoS alarms indicate a decrease in service quality.

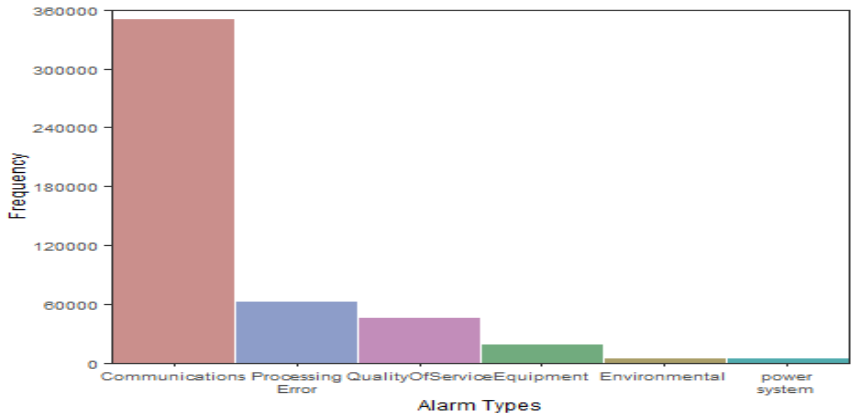


*Figure 3.3 Alarms categorized by alarm types*

Figure 3.4 shows the specific problems that result from the failures versus their frequency of occurrence on a logarithmic scale. SCTP link fault and user plane path fault alarms resulting from transmission faults appear the most in the dataset.

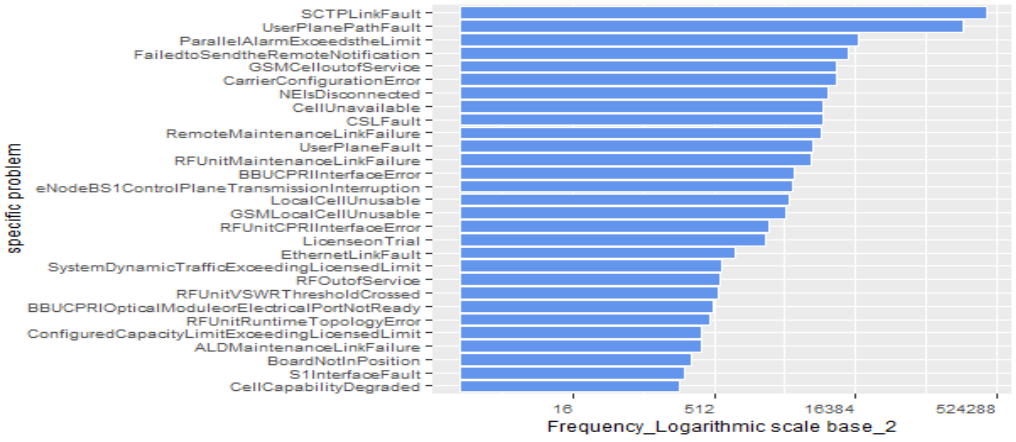


*Figure 3.4 Alarms categorized by probable cause*

To describe some of the specific problems in the dataset, The SCTP link fault alarm appears when the base station detects that the SCTP connection can't handle service, which means that the link

is only available in one way, receiving or sending, and cannot receive and send at the same time. RF Unit Maintenance Link Failure: When the BBU does not receive a response from an RF unit, this alarm is triggered. If any of the following components are defective, this alarm will be triggered: CPRI optical fibers, CPRI cables, optical modules, or board hardware. The RF unit's services are disrupted, and the RF unit can't be maintained remotely. Table 3.1 lists example of probable causes with their frequency of occurrence in the dataset.

| BTS Alarms/probable cause | Frequency |
|---|---|
| SCTP Link Fault | 399881 |
| User Plane Path Fault | 231050 |
| GSM Cell out of Service | 10337 |
| Carrier Configuration Error | 10316 |
| NE Is Disconnected | 8186 |
| Remote Maintenance Link Failure | 6960 |
| RF Unit Maintenance Link Failure | 5488 |
| BBU CPRI Interface Error | 3666 |

*Table 3.1 Most common BTS alarms with their frequency of occurrence*

### 3.1.3 Data Cleaning

Alarm data contains alarm events indicating an alert on a problem occurring on a network element. In Figure 3.5, sequential events from Oct. 2020 to Feb. 2021 are shown. The alarm event versus the date corresponding to the time stamp for all events in this period. There are 276,178 different event types in the data set. As can be seen, there is a strong correlation between these events. Rather than this correlation, the dataset contains chattering alarms. The most common type of nuisance alarm is one that repeats itself repeatedly over a short period of time.

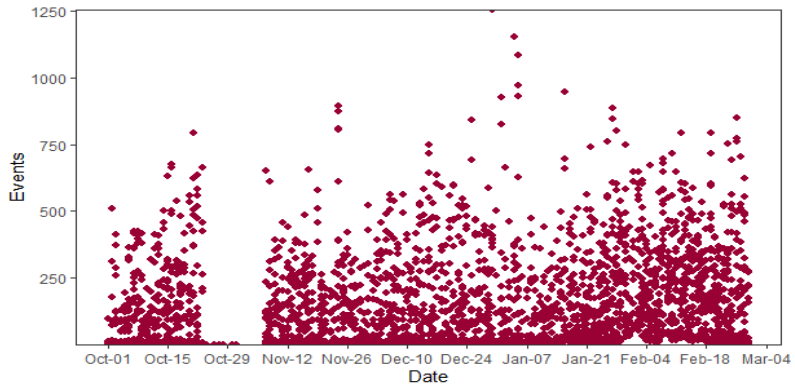


*Figure 3.5 Alarm event versus date*

An alarm that is activated and cleared three or more times in 1 minute is often used as a preliminary identification for the worst chattering alarms [42]. These alarms are removed from the dataset after representing them with one instance of the alarm. Also, alarms from other devices are removed as only BTS alarms are considered for this work. Irrelevant fields and missing values are also cleaned up from the dataset.

### 3.1.4 Data Normalization

Besides, from Figure 3.2, it is clear that the dataset is strongly unbalanced, as 86.5% of the overall alarm severity belongs to major alarms. The model's training is skewed since the model lacks sufficient samples of the minority class to appropriately recognize a pattern. Although it is not able to distinguish items that should belong to the minority class, a random model that always predicts the majority class can achieve better results. As a result, having a balanced dataset is a standard technique in many machine learning models, allowing for a better understanding of the model's performance. The alarm dataset contains sufficient samples so the major class is reduced to balance the minority class.

## 3.2 Alarm pattern mining

Pattern mining techniques have been used to analyze telecommunication data and find various types of patterns for fault management. Going to the original problem and the methods used for this work are given these preliminaries for finding patterns. Because the goal is to find regular patterns, the knowledge discovery approach is used to accomplish it. Also the problem is based on a market basket analysis, the first step is to identify items that we want to investigate and then define transactions for them. Then we'll go on to selecting and visualizing the rules that emerged from the mining process.

### 3.2.1 Matrix of transactions

To use any rule mining approach, we must first convert the data from a frame format to transactions, where each row represents a transaction and each column represents an item. While the data mining process extracts associations in the form of A implies B or in the form of "IF-THEN" statements, we need to aggregate alarm features to form a two-dimensional dataset.

The alarm data has many categorical features. In this research, alarms are represented by their severity, alarm name, occurrence time, and alarm source. Therefore, only four features are selected and the three features are aggregated to form a one-dimensional representation, which is a combined feature. Let's call this new feature with the timestamps [timestamp, new feature]. In the market basket analysis, the items are alarms, so in the transaction matrix the alarms with the new features are the items in the horizontal direction while the timestamp indicating the transaction id is shown in the vertical direction of the transaction matrix. Alarms which exist at the given timestamp are assigned 1, while alarms which don't occur at that given time will be assigned zero value. Alarms occur simultaneously, so to calculate the number of transactions for the five-month dataset, a three-hour time bin is selected randomly.

- The number of transactions in October, (31*24)/3 = 248
- The number of transactions in November, (30*24)/3 = 240
- The number of transactions in December, (31*24)/3 = 248
- The number of transactions in January, (31*24)/3 = 248
- The number of transactions in February, (28*24)/3 = 224

For the purpose of extracting association rules data must be converted into one hot representation or binary form. This is performed in excel with the help of a pivot table. Table 3.2 shows how this transactional dataset is constructed.

| Timestamp | Alarm-1 | Alarm-2 | Alarm-3 | Alarm-4 | Alarm-5 |
|---|---|---|---|---|---|
| Tid1 | 0 | 1 | 1 | 0 | 1 |
| Tid2 | 0 | 0 | 0 | 0 | 1 |
| Tid3 | 1 | 1 | 0 | 1 | 0 |
| Tid4 | 0 | 1 | 0 | 1 | 1 |
| Tid5 | 1 | 0 | 0 | 0 | 1 |
| Tid6 | 0 | 0 | 1 | 1 | 0 |
| Tid7 | 0 | 0 | 1 | 0 | 0 |

*Table 3.2 An example of alarm transaction*

### 3.2.2 Rule generation

The transaction dataset now shows a matrix of alarms that have been observed together. We have no idea how often they are seen together, and we have no idea what the rules are. For the purpose of mining the rules, the Rapidminer operators are linked together. The FP-Growth operator produces frequent items that can be used to generate association rules, but first the parameters of the rules must be defined. The output of the Create Association Rules Operator includes a summary of rules as well as data on the total number of items mined and the minimum parameters set. The following minimum parameters are set, and other parameters are set to Rapidminer default.

- Minimum support threshold: 0.1
- Minimum confidence threshold: 0.8

| Premise | Conclusion | Conclus | Confide | Convict | Gain | Laplace | Lift | Ps | Total Support |
|---|---|---|---|---|---|---|---|---|---|
| [Minor#External Cl | [Major#RF Unit Ma | 1.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.9 | .1 | .1 |
| [Minor#Remote Ma | [Major#RF Unit Ma | 1.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.9 | .1 | .1 |
| [Minor#Remote Ma | [Major#RF Unit Ma | 1.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.9 | .1 | .1 |
| [Minor#External Cl | [Majorr#RF Unit V | 2.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.9 | .1 | .1 |
| [Minor#RF Unit VS | Major#BBU CPRI | 1.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.7 | .1 | .1 |
| [Major#Local Cell | Major#BBU CPRI | 2.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.7 | .1 | .1 |
| [Minor#RF Unit VS | Major#BBU CPRI | 1.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.7 | .1 | .1 |
| [Minor#RF Unit VS | Major#BBU CPRI | 2.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.7 | .1 | .1 |
| [Minor#External Cl | Major#BBU CPRI | 1.0 | 1.0 | 20.7 | -.1 | 1.0 | 15.7 | .1 | .1 |

*Table 3.3 A sample of association rule*

These parameters depend on the interest of the researcher and can be set in a different setup. As we increase the minimum thresholds, the total number of association rules extracted will be smaller and vice versa. The execute script and write to excel operators help to write the result of the association rule in excel format. Table 3.3 shows the rules extracted, the premise, and the conclusion with respect to the parameters for that rule. The detailed description of the rules is not described here, rather it is important for domain experts only.

### 3.2.3 Rule Visualization

Following the association of the rules, now it is possible to visualize them using measurements of interest. After that, we can easily select interesting rules and visualize them over time. A scatter plot of lift vs support and confidence (Figure 3.6) is used to represent the whole set of rules. Most

of the rules have a lift above eight, with 10% support and above 80% confidence.

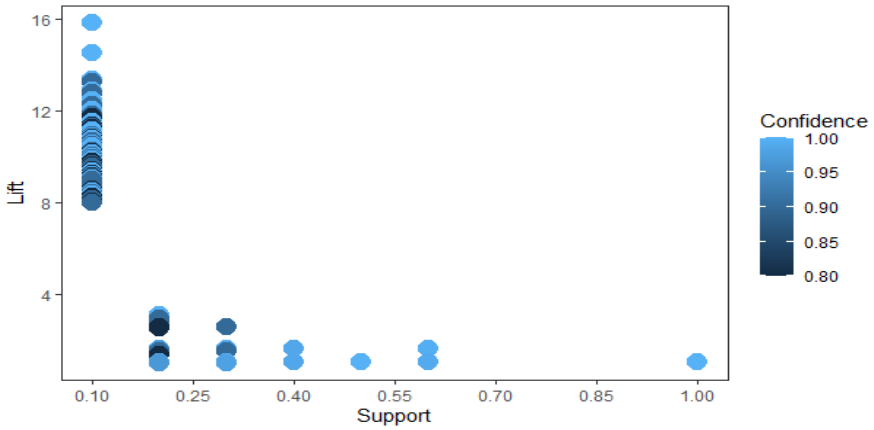


*Figure 3.6 Scatter plot of rules*

The more lift values we have, the greater the probability of an alarm occurring, and for this experiment, the maximum lift value obtained is 15.9, which is a good value**.** The rules drawn from the association analysis help domain experts to reformulate correlation patterns as well as to study network behavior, as these rules show the current status of the network. All rules are not always important, so it is necessary for domain experts to select relevant rules which facilitate their work. In the next chapter, selected rules from the association rule mining process will be used for training the three deep learning algorithms for predicting BTS alarms.

# 4. Model Training and Evaluation

## 4.1 Deep Learning Model Training

In the above section we have extracted sequential patterns from historical alarm data by the help of data mining. Now this knowledge is used to train the three deep learning models, LSTM, Bi-LSTM, and GRU, that are available within the Keras framework to predict the target events. The three models are analyzed in order to select the best performing model for the alarm prediction task. Alarms are in word text format containing different categorical information about the affected device, and this textual data must be represented in vector form in order to feed the neural network. The use of word embedding by training and pretrained word embedding, two ways of feature representation to feed embedding to the neural network, are evaluated. The models are established in Tensorflow python.

The alarm dataset used for this analysis undergoes a data mining process. The process extracts frequent patterns from the transaction dataset, then from the frequent patterns, the association rules are extracted. In the Rapiminer studio, the pre-built models are represented by the FP-Growth operator, which performs frequent pattern mining using the FP-Growth algorithm, and the operator that creates association rules, extracts association rules, and performs the datamining task with the help of other operators. The other operators are attribute selection, numerical to binomial converter, execute script and write to excel. The output of association rules generates around 300k rules. This result contains many irrelevant rules, so we have to select rules which are not deterministic for the domain experts. The association rule metrics discussed in the above sections are used as criteria to select relevant once. In this research, manly equipment alarms are selected as target alarms based on that 15,300 rules are used to constrict the corpus and the corpus has 176 unique tokens. The text data (corpus) is preprocessed using python. First, the corpus follows a tokenization process which involves splitting the bigger data into smaller segments. Next, the text is converted to sequences. In this way, we can interpret the text data into numbers for better analysis. Then training data is created with "X", the independent variable containing input text, and "Y", the dependent variable which contains the target alarms.

### 4.1.1 Hyperparameter setup

Hyperparameters determine the network structure and how the network is trained. As the model performance is directly affected by these parameters, it is important to carefully select the best combination of hyperparameters.

| Parameter | Value |
|---|---|
| Embedding dimensions | 128 |
| Batch size | 64 |
| Learning rate | 0.01 LSTM & BiLSTM, 0.001 for GRU |
| Optimizer | Adam |
| Output activation function | softmax |
| Loss function | Categorical cross entropy |
| Input and hidden layer activation | ReLU |
| Number of Epochs | 100 |

*Table 4.1 Hyperparameter used*

All the three deep learning models (LSTM, Bi-LSTM, GRU) are implemented in python. LSTM and Bi-LSTM use the same number of layers while the number of input layers for the GRU is different. The input layer is an embedding layer with the purpose of mapping the input text to its vector representation or the selected word embeddings. ReLU activation is used for the input and hidden layer. The last layer is a dense layer with a softmax activation function.

The models are trained with a batch size of 64 and an epoch of 100. It's the number of times that the models are exposed to the training dataset. Early callback functionality is implemented to stop the training process once the network performance stops improving. Categorical cross entropy and Adam are used as a loss function and optimizer respectively. A learning rate of 0.01 used for LSTM and Bi-LSTM. For the GRU-based model, 0.001 is applied as a learning rate. All of the parameters are set based on model performance during model tuning. The experiment was conducted on an 8GB RAM Lenovo laptop with no GPU.

### 4.1.2 Training using domain trained Word Embedding

The proposed deep learning models (LSTM, Bi-LSTM, GRU) were tested using domain-trained word embedding. Alarm patterns or the knowledge base patterns construct the corpus. In this task, a corpus is a collection of texts, and an alarm pattern is a collection of BTS communication, equipment, QoS, power, environmental, and process alarms. The domain-trained word embedding uses the corpus which is constructed from the alarm data and associates a vector with it.

### 4.1.3 Training using Pretrained Word Embedding

The second test implements pretrained word embedding representation with Word2vec. Genism, a python library, is used for the Word2vec algorithm. The corpus is a Googelnews dataset that is available online. The dataset contains more than one billion words, but there is a limit on domain-specific words, which results in out of vocabulary of words.

## 4.2 Model Evaluation

The model evaluation contains a combination of the deep learning models with two different text representation methods. The below setups are used to evaluate the performance of the models.

- LSTM-based model with domain-trained word embedding.
- LSTM-based model with pretrained word embedding.
- Bi-LSTM-based model with domain-trained word embedding.
- Bi-LSTM-based model with pretrained word embedding.
- A GRU-based model with domain-trained word embedding.
- GRU-based model with pretrained word embedding.

The results of these configurations are shown in Table 4.2, in terms of test accuracy and loss. In addition to the above configurations, a test is conducted with different setups to study the effect of embedding dimension and batch size on model performance. Table 4.3 and 4.4 show the results obtained. The alarm prediction Accuracy as a metric is the ratio of correct predictions to all the predictions made.

| Word Embeddding Techniques | LSTM | | Bi-LSTM | | GRU | |
|---|---|---|---|---|---|---|
| | Test Accuracy | Test loss | Test Accuracy | Test loss | Test Accuracy | Test loss |
| Domain trained word embedding | 0.89 | 0.1037 | 0.93 | 0.0586 | 0.88 | 0.0897 |
| Pretrained word embedding | 0.88 | 0.1027 | 0.92 | 0.0709 | 0.90 | 0.0691 |

*Table 4.2 Test accuracy and loss with corresponding word embedding techniques*

Testing the three deep learning models, LSTM, Bi-LSTM, and GRU, implements two different feature representation techniques. The models are first tested using a domain-trained word embedding technique, meaning the corpus being built in this case is from a sequence of alarms which are constructed from the knowledge discovery process. The second test is conducted with the help of a pretrained word embedding technique, building its corpus from the Googelnews dataset.

| Embedding dimention | LSTM | | | Bi-LSTM | | | GRU | | |
|---|---|---|---|---|---|---|---|---|---|
| | Number of Epoch | Test Accuracy | Test loss | Number of Epoch | Test Accuracy | Test loss | Number of Epoch | Test Accuracy | Test loss |
| 16 | 29 | 0.88 | 0.0899 | 29 | 0.92 | 0.0686 | 50 | 0.79 | 0.1526 |
| 32 | 29 | 0.89 | 0.1016 | 32 | 0.92 | 0.0793 | 50 | 0.83 | 0.1236 |
| 64 | 25 | 0.88 | 0.0912 | 31 | 0.92 | 0.0679 | 50 | 0.85 | 0.1017 |
| 128 | 24 | 0.89 | 0.0949 | 33 | 0.93 | 0.0586 | 46 | 0.88 | 0.0803 |
| 256 | 22 | 0.89 | 0.0984 | 21 | 0.91 | 0.0727 | 50 | 0.85 | 0.1073 |
| 300 | 16 | 0.90 | 0.0947 | 29 | 0.89 | 0.0928 | 50 | 0.89 | 0.0822 |

*Table 4.3 Test accuracy and loss with corresponding embedding dimension*

| batch size | LSTM | | | Bi-LSTM | | | GRU | | |
|---|---|---|---|---|---|---|---|---|---|
| | Number of Epoch | Test Accuracy | Test loss | Number of Epoch | Test Accuracy | Test loss | Number of Epoch | Test Accuracy | Test loss |
| 16 | 12 | 0.82 | 0.1584 | 14 | 0.86 | 0.1604 | 49 | 0.85 | 0.1014 |
| 32 | 19 | 0.87 | 0.1149 | 21 | 0.90 | 0.0911 | 98 | 0.90 | 0.0770 |
| 64 | 26 | 0.89 | 0.0872 | 27 | 0.93 | 0.0589 | 47 | 0.86 | 0.0993 |
| 128 | 30 | 0.89 | 0.1017 | 51 | 0.92 | 0.0708 | 63 | 0.86 | 0.0940 |
| 256 | 44 | 0.88 | 0.0822 | 40 | 0.92 | 0.0724 | 99 | 0.86 | 0.0973 |

*Table 4.4 Test accuracy and loss with corresponding batch size*

A test was also conducted to study the effect of parameters on the model performance. These parameters are crucial in deep learning algorithms. Different embedding dimensions and batch sizes are implemented on LSTM, Bi-LSTM, and GRU. The test is measured by model accuracy, loss, and the number of epochs.

# 5. Result and Discussion

## 5.1 LSTM based model Result

The LSTM based model is trained with two types of word embedding techniques: domain-trained word embedding and pretrained word embedding. The model shows good performance in both techniques, and domain-trained word embedding achieved slightly higher performance than pretrained word embedding. While the number of input layers for LSTM and GRU is different, in the given embedding dimension and batch size when the LSTM model is compared with the GRU-based one, the LSTM-based model scores better than the GRU. This better performance is achieved in the case of domain trained word embedding with an accuracy score of 89%. However, the accuracy obtained from the GRU model for pretrained word embedding gives a better performance than the LSTM.

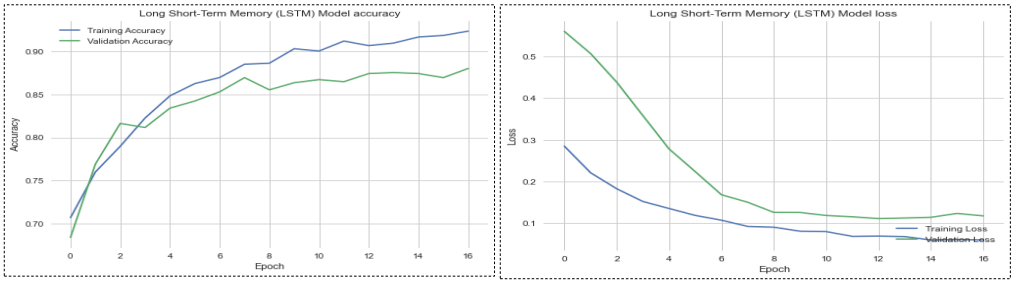


*Figure 5.1 LSTM model using word embedding learning curve*

The learning curve of the LSTM model with domain-trained word embedding is shown in Figure 5.1. The model converges after 16/100 epochs with early callback functionality as a regularization strategy.

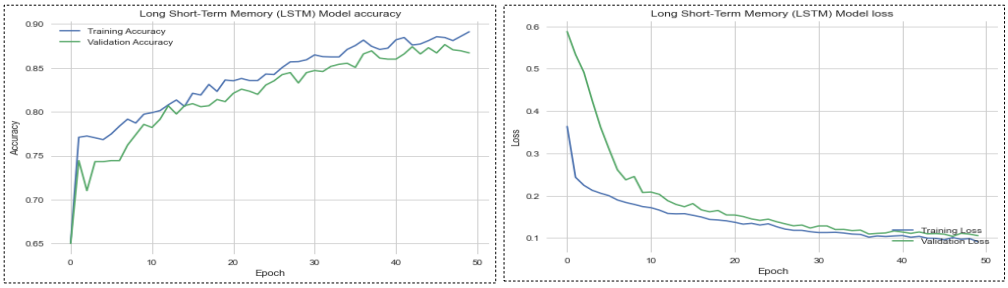


*Figure 5.2 LSTM model using pertained word embedding learning curve*

While the LSTM model is trained with pretrained word embedding, its loss actually moves towards a minima after 50/100 epochs. Figure 5.2. The second approach takes a longer number of epochs (the number of times that the model is exposed to the training dataset). But the training time is much faster in the case of pretrained word embedding.

## 5.2 Bi-LSTM based model Result

The Bidirectional LSTM outperforms both the LSTM and GRU-based models. This indicates its adequacy for alarm prediction tasks. Two ways of word embedding are employed: domain-trained word embedding and pretrained word embedding. The evaluation shows that both techniques show a good performance in representing the alarms. The Bidirectional LSTM trained with both word embedding techniques has better prediction performance than the LSTM and GRU models, with an accuracy of 93% for domain trained and 92% with pretrained word embedding.

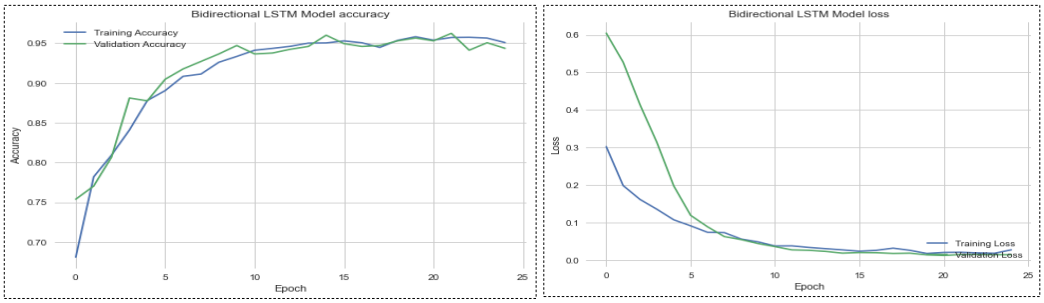


*Figure 5.3 Bidirectional LSTM model using word embedding learning curve*

The Bi-LSTM model trained with domain-trained word embedding converges to some minima with an epoch of 25/100 as shown in Figure 5.3. Also, Figure 5.4 shows an early callback functionality that is implemented to stop the training process at an epoch of 25 for pretrained word embedding.

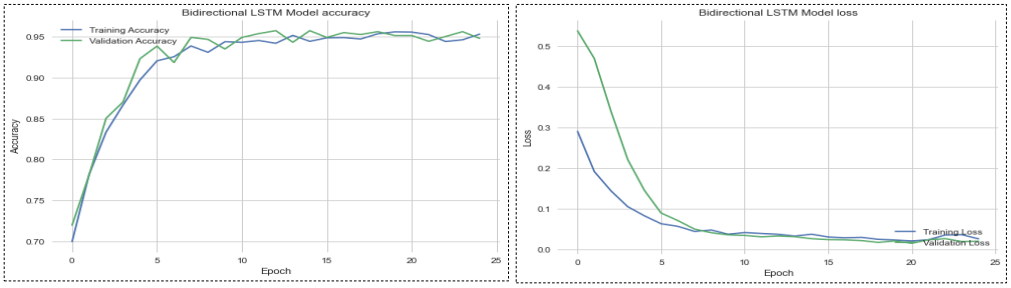


*Figure 5.4 Bidirectional LSTM model using pertained word embedding learning curve*

## 5.3 GRU based model Result

In this model, a good result is found by the pretrained word embedding for the alarm prediction purpose, with an accuracy of 90%. The test loss measured in this model is also slightly lower than that of LSTM. This model shows its suitability for the alarm prediction task next to the LSTM-based model. Despite the prediction performance, the GRU-based model is faster in model training.

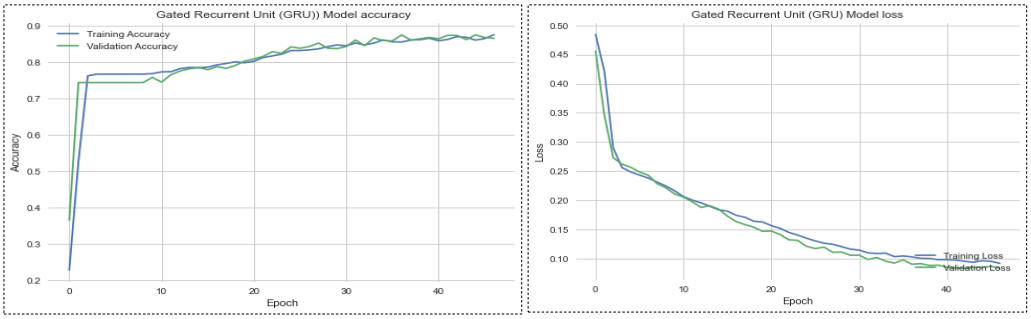


*Figure 5.5 GRU model using word embedding learning curve*

In Figure 5.5, training the model with domain trained word embedding comes to a minimum loss after 40 epochs. From tables 4.2 and 4.3, it is clear that the model takes longer epochs than the other two models. Also, Figure 5.6 shows model training with pretrained word embedding which converges after 50 epochs.

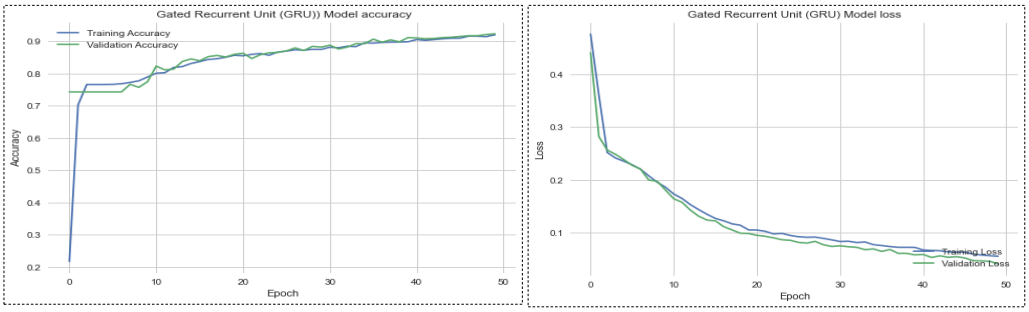


*Figure 5.6 GRU model using pertained word embedding learning curve*

## 5.4 Discussion

Three deep learning models are presented: LSTM, Bi-LSTM, and GRU, each with two feature representation techniques. The different feature representing techniques help in improving the

model performance and training speed. These models achieve better performance with a batch size of 64 and an embedding dimension of 128. The Bi-LSTM model outperforms the LSTM and GRU models in both features, representing methods with accuracy as a criterion. The LSTM has slightly higher performance than GRU in the case of domain-trained word embedding, and GRU has become better in pretrained word embedding. While training the models, the GRU has a higher training speed than the other two. Overall, the three models can be used for alarm prediction tasks in both feature representation techniques.

When domain-trained word embedding is compared to pretrained word embedding, the first technique outperforms the latter, which could be because domain-specific words with no common representation in natural language processing lead to an out of vocabulary of words when trained with pretrained word embedding. In most cases, the pretrained word embedding has better performance than word embedding by training approaches in common NLP. However, in the case of alarm prediction practical applications, it is recommended to use domain-trained word embedding as it can help in representing each word in its vector form. The difference in performance between these two approaches comes from the loss of information as a result of the lack of vocabulary issue. The pretrained word embedding lacks a full vector representation for the alarm corpus, but it has a faster training process as compared to domain-trained word embedding, which has a much slower training process. The different embedding dimensions and batch sizes have a direct effect on model training speed. As the embedding dimension increases, model training becomes slower and vice versa. The same is true for batch size. Whenever the batch size becomes smaller, the training speed becomes very slow. Most of the time, the changes in these parameters do not show a significant change in model performance.

# 6. Conclusion and Future Works

## 6.1 Conclusion

Nowadays, telecommunication networks occupy a central position in our world. Indeed, they allow us to share worldwide a huge amount of information. Networks are, however, complex systems, both in size and technological diversity. Therefore, it makes their management and reparation more difficult. In order to limit the negative impact of such failures, some mechanisms have to be developed to detect a failure whenever it occurs, analyze its root causes to solve it efficiently, or even predict this failure as prevention is better than cure.

As can be seen in statement problem section 1.1, there are a lot of incidents registered in the fault management system, and to restore these incidents takes a very long time. A decline in QoS as well as a loss of revenue are some of the outcomes. To overcome this issue, predictive monitoring or predicting alarms in fault management will help to identify faults before they result in a total hardware failure.

The result of this research indicates that the three deep learning models, LSTM, Bi-LSTM, and GRU, can be used for alarm prediction tasks. The models have satisfactory generalization and the accuracy for alarm prediction can reach about 93% with a loss of 0.0586 in the Bi-LSTM based model. While the two-word embedding techniques show a slight change in model performance domain, trained word embedding is a suitable approach for representing alarms to their corresponding vector representations as it overcomes the problem of words out of vocabulary.

## 6.2 Future Works

The research study area has many challenges with many subsystems talking to each other, leading to a failure which can't be easily detected, and the different data formats make the data preprocessing work a difficult task. Data collection, preprocessing, mining alarm logs, and predictive models are important topics for alarm prediction purposes, which is left for future work also. This research uses a data mining approach to find useful patterns from alarm logs with the help of the FP-Growth algorithm. There are other algorithms for this purpose, so it is better to test these algorithms with different data preprocessing methods. This research uses deep learning

models like LSTM, Bi-LSTM, and GRU for training these patterns. Other models with different model configurations are interesting areas to be studied. The below ideas are suggestions to be dealt with in the future.

- Rather than BTS alarms, there are other types of alarms from different subsystems that exist in the fault management system, so it is better to consider the alarm prediction in these areas.
- Testing other alarm prediction approaches for different areas of implementation will help in finding a suitable method.
- It is also important to consider the data preprocessing, data mining algorithm and model tuning while improving this alarm prediction work.

# Reference


[1] Abiad, Mohammad; Kadry, Seifedine; Ionescu, Sorin, "Preventive & Predictive Maintenance of Telecommunication Equipment - A Review,", *[IEEE 2018 4th International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT) - Mangalore, India (2018.9.6-2018.9.8)] 2018 4th International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT) ,154–159.* 2018.

[2] Ethio telecom "Ethio telecom Incident report," ,2020.

[3] Klemettinen, M., Mannila, H. & Toivonen, H. "Rule Discovery in Telecommunication Alarm Data," *Journal of Network and Systems Management* 7, 395–423 (1999).

[4] D. Medved, K. Brlas; D. Saric (1998). "Fault analysis and prediction in telecommunication access network,", 1998.

[5] Bodhisattwa Gangopadhyay, Artur Arsenio, Claudia Antunes, "Comparative Study of Pattern Mining Techniques for Network Management System Logs for Convergent Network,", 2012.

[6] García, A. J., Toril, M., Oliver, P., Luna-Ramírez, S., & Ortiz, M, ''Automatic alarm prioritization by data mining for fault management in cellular networks,'' *A review Expert Systems with Applications, Spain,* 2020.

[7] G. Zargarian, "Unsupervised Machine Learning for Mining Alarm Logs of a Large Telecommunication Network", Msc in Communications and Computer Networks Engineering ,2018.

[8] Wrench C., Stahl F., Le T., Di Fatta G., Karthikeyan V., Nauck D. (2016), "A Method of Rule Induction for Predicting and Describing Future Alarms in a Telecommunication Network,", In: Bramer M., Petridis M. (eds) Research and Development in Intelligent Systems. 2016.

[9] Caravela, Ivan; Arsenio, Artur; Borges, Nuno (2016). "A Closed-Loop Automatic Data-Mining Approach for Preventive Network Monitoring,", *Journal of Network and Systems Management,* 2016.

[10] Hatonen, K.; Klemettinen, M.; Mannila, H.; Ronkainen, P.; Toivonen, H. (1996). "TASA: Telecommunication Alarm Sequence Analyzer or how to enjoy faults in your network.",

*[IEEE NOMS '96 - IEEE Network Operations and Management Symposium - Kyoto, Japan (15-19 April 1996)] Proceedings of NOMS '96 - IEEE Network Operations and Management Symposium,* 1996.

[11] Gary M. Weiss, "Predicting Telecommunication Equipment Failures from Sequences of Network Alarms, AT&T Labs, Fordham University, " July 2000.

[12] Imran Khan, J. Huang, N. Tung, "Learning Time-based Rules for Prediction of Alarms from Telecom Alarm Data Using Ant Colony Optimization," International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 03 – Issue 01, January 2014.

[13] Jakobson, G.; Weissman, M. (1993). "Alarm correlation," *IEEE Network,*1993.

[14] Cherrared, Sihem; Imadali, Sofiane; Fabre, Eric; Gossler, Gregor; Yahia, Imen Grida Ben (2019). "A Survey of Fault Management in Network Virtualization Environments: Challenges and Solutions,*" IEEE Transactions on Network and Service Management,*2019.

[15] Mika Klemettinen; Heikki Mannila; Hannu Toivonen (1999). "Rule Discovery in Telecommunication Alarm Data,*"* 1999.

[16] Abraham, Ajith; Gandhi, Niketa; Pant, Millie (2019). "Design and Implementation of a Fault Management System,*" [Advances in Intelligent Systems and Computing] Innovations in Bio-Inspired Computing and Applications Volume 939 (Proceedings of the 9th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2018) held in Kochi, India during December 17-19, 2018).*

[17] Nouioua M., Fournier-Viger P., He G., Nouioua F., Min Z. (2021) "A Survey of Machine Learning for Network Fault Management,*"* 2021.

[18] HUAWEI TECHNOLOGIES CO., LTD, eRAN Troubleshooting Guide, 2016.

[19] Keerthi Sumiran, "An Overview of Data Mining Techniques and Their Application in Industrial Engineering,*"* Asian Journal of Applied Science and Technology (AJAST) (Open Access Quarterly International Journal) Volume 2, Issue 2, Pages 947-953, 2018.

[20] Margaret H Dunham, Dr.Sridhar Seshadri. "Data Mining- Introductory and Advanced Topics,*"* January 2006.

[21] Loshin, David (2013), "Knowledge Discovery and Data Mining for Predictive Analytics*,"* 2013.

[22] Ansari, Sohaib Zafar, "Market basket analysis : trend analysis of association rules in different time periods,*"* 30-Jul-2019, Dissertation presented as the partial requirement for obtaining a Master's degree in Statistics and Information Management, specialization in Marketing Research e CRM.2019.

[23] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, "Introduction to Data Mining,*"* book, 2005.

[24] Dr. Hui Xiong, "Association Analysis: Basic Concepts and Algorithms,*"* Rutgers University, 2006.

[25] Gordon S. Linoff, Michael J. A. Berry, "Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management,*"* third edition. 2011by Wiley Publishing, Inc. Indianapolis, Indiana, 2011.

[26] Ms.P. Nithya, M.Sc., M.Phil.,2K. Sivapriya, 3M. Rajshree, "An Overview of Market Basket Analysis using Aprirori Algorithm,*"* International Journal of Advance research in science and engineering, vol.no9,2020.

[27] Akinbinu, "A.: New telecommunication technologies and the financial health of the Nigeria telecommunications Ltd,*"* Nigerian J. Eng. Manag. 14–22 (2001).

[28] Jiawei Han; Hong Cheng; Dong Xin; Xifeng Yan (2007). "Frequent pattern mining: current status and future directions,*"* 2007.

[29] Nasreen, Shamila; Azam, Muhammad Awais; Shehzad, Khurram; Naeem, Usman; Ghazanfar, Mustansar Ali (2014). "Frequent Pattern Mining Algorithms for Finding Associated Frequent Patterns for Data Streams: A Survey." *Procedia Computer Science,*

2014.

[30] J.-P. Vasseur, M. Pickavet, and P. Demeester, "Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS," Elsevier, 2004.

[31] Yu, Yong; Si, Xiaosheng; Hu, Changhua; Zhang, Jianxun (2019). "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," Neural Computation, 2019.

[32] Dessì, D.; Reforgiato Recupero, D.; Sack, H. "An Assessment of Deep Learning Models and Word Embeddings for Toxicity Detection within Online Textual Comments. Electronics," 2021.

[33] Jang, Beakcheol; Kim, Myeonghwi; Harerimana, Gaspard; Kang, Sang-ug; Kim, Jong Wook (2020). "Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism," *Applied Sciences,* 2020.

[34] Shrestha, Ajay; Mahmood, Ausif (2019). "Review of Deep Learning Algorithms and Architectures," *IEEE Access,* 2019.

[35] Apaydin, Halit; Feizi, Hajar; Sattari, Mohammad Taghi; Colak, Muslume Sevba; Shamshirband, Shahaboddin; Chau, Kwok-Wing (2020). "Comparative Analysis of Recurrent Neural Network Architectures for Reservoir Inflow Forecasting," *Water,*2020.

[36] PijiLi, "Optimization Algorithms for Deep Learning," Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, 2019.

[37] I. Goodfellow, B. Yoshua, and A. Courvill, "Deep Learning," MIT press, 2016.

[38] Amitha Mathew, P. Amudha and S. Sivakumari, "Deep Learning Techniques: An Overview," January 2021.

[39] O Maimon , L Rokach "Introduction to knowledge discovery in databases Data mining and knowledge discovery handbook, " 2005.

[40] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth, "From Data Mining to Knowledge Discovery in Database," 1996.

[41] Mirela Danubianu, "Step by Step Data Preprocessing for Data Mining. A Case Study," Proceedings of the International Conference on Information Technologies 2014.

[42] Cai, Shuang, Palazoglu, Ahmet, Zhang, Laibin, Hu, Jinqiu (2018). "Process alarm prediction using deep learning and word embedding methods." *ISA Transactions,*2018.

[43] Alfiqra and A U Khasanah, "Implementation of Market Basket Analysis based on Overall Variability of Association Rule (OCVR) on Product Marketing Strategy, *"* IOP Conference Series: Materials Science and Engineering, Volume 722, 3rd International Conference on Engineering Technology for Sustainable Development (ICET4SD) 23–24 October 2019.

[44] Dr.R. Shankar1 and Dress. Duraisamy, "Analysis of Data Mining Tasks, Techniques, Tools, Applications and Trends," IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 20, Issue 5, Ver. II (Sep - Oct 2018), PP 12-19, 2018.

[45] Bharati M. Ramageri, "Data Mining Techniques and Applications," *Indian Journal of Computer Science and Engineering Vol. 1 No. 4 301-305,* 2010.

[46] Bengio, Y.; Simard, P.; Frasconi, P. "Learning long-term dependencies with gradient descent is difficult," IEEE Trans. Neural Newts. 1994, 5, 157–166.

[47] Jonathan Lorraine and David Duvenaud. "Stochastic hyperparameter optimization through hypernetworks." arXiv preprint arXiv:1802.09419, 2018.

[48] Anil K Jain. Data clustering: "50 years beyond k-means. Pattern recognition letters," 31(8):651–666, 2010.

[49] Toshihiro Takahashi. "Statistical max pooling with deep learning," July 3 2018. US Patent 10,013,644.

[50] Chaochun Liu, Yaliang Li, Hongliang Frei, and Ping Li. "Deep skip-gram networks for text classification," In Proceedings of the 2019 SIAM International Conference on Data Mining, pages 145–153. SIAM, 2019.