Detection of Online Phishing Email using Dynamic Evolving Neural Network Based on Reinforcement Learning

Sami M. Smadi

PhD

2017

Detection of Online Phishing Email using Dynamic Evolving Neural Network Based on Reinforcement Learning

Sami M. Smadi

A thesis submitted in partial fulfilment
of the requirements of the
University of Northumbria at Newcastle
for the degree of Doctor of Philosophy

Research undertaken in the Faculty of
Engineering and Environment

March 2017

Abstract

Phishing has been the most frequent cybercrime activity over the last 15 years and has caused billions of dollars to be stolen. This happens due to the fact that phishing attackers always use new (zero-day) and sophisticated techniques to deceive online customers. The most common way to initiate a phishing attack is by using email. In this thesis, a novel framework is proposed that combines a neural network with reinforcement learning for detecting online phishing attacks.

This thesis addresses the effectiveness of phishing email detection, and it makes the following contributions. Firstly, a novel pre-processing system has been designed to gather and extract the features and patterns of phishing email. To cover all behaviour that phishers use to deceive online customers, fifty features were selected. Pre-processing is divided into three layers, based on the main types of email content. Secondly, a novel algorithm has been proposed for the exploration of new phishing behaviour. The proposed algorithm has the ability to select the effective list of features from the list of features extracted in the pre-processing phase. Thirdly, this thesis proposed a novel Dynamic Evolving Neural Network using Reinforcement Learning (DENNuRL) algorithm, which can be used to generate the best neural network for classification problem based on reinforcement learning idea. Finally, a novel framework for Phishing Email Detection System (PEDS) has been proposed. The PEDS has the ability to adapt itself to generate a new PEDS that reflects changes in behaviour. Therefore, reinforcement learning is adopted in the proposed framework combined with neural network to enhance the system dynamically over time in the online mode. The proposed technique can effectively handle zero-day phishing attacks.

The proposed phishing email detection model was trained, tested and validated in the online mode using an approved dataset. The promising results showed that the DENNuRL can provide an effective means of phishing detection. The proposed model successfully classified and identified approximately 98.6% of phishing emails selected from the test dataset, with low false positive rates of 1.8%. A comparison with other similar techniques using the same dataset shows that the proposed technique outperforms the existing methods.

Declaration

I declare that the work contained in this thesis has not been submitted for any other award and that it is all my work. I also confirm that this work fully acknowledges opinions, ideas and contributions from the work of others.

Sami Smadi

March 2017

Acknowledgements

First of all, I would like to thank God, ALLAH, whose grace has led me to this important moment of my life. I am truly thankful and appreciative of my supervisors for their valuable contributions and continuing efforts towards the success of this thesis. Without a doubt, none of this work would have been possible without their help. Personal thanks go to my supervisor Dr. Nauman Aslam, who always appreciated and had a vision for the bigger picture of this project, encouraging me to combine the ample amount of research available worldwide to form a better understanding of the study. I also thank him, for the continuous advice, support, and valuable scientific input. I also want to thank my supervision committee members Dr. Rafe Alasem and Dr. Li Zhang for advice, management of this study, reviewing my thesis, and providing constructive comments that help me improve my work. As well, I honoured to have been supervised by Professor Alamgir Hossain for his excellent guidance throughout the preliminary stages of my research. Also, I thank my colleagues, PhD researchers of LAB F7 and LAB F6 for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years. Also, I acknowledge all my colleges in the college of science Al-Zulfi, KSA. In addition, my work has benefited from and has been influenced by discussions with a number of people over the years. For these discussions, I express my gratitude to my friends Dr. Mohammad Alauthman, Dr. Majed Alsanea and Dr. Mohammad Alalaween. They offered a lot of help and advice on my research and wonderful friendship. My

deepest appreciation goes to all my friends, who are many and whom each has offered me unique encouragement and support every step of the way. Finally, my thanks and affection go to my dear parents, my greatest source of love and affection and my biggest support system. To my brothers, the providers of an endless amount of friendship and encouragement throughout this journey. Last but not the least, I am also thankful to my wife for always being there for me in the ups and downs of my life and tremendous support.

Contents

List of Figures

List of Tables

Acronyms

AIM AOL Instant Messenger

ANN Artificial Neural Network

AOL America Online

APTs Advanced Persistent Threats

APWG Anti-Phishing Working Group

AUC Area Under the ROC curve

AMGA Adaptive Merging and Growing Algorithm

CA Constructive approach

CART Classification And Regression Tree

CPA Constructive-pruning approach

CSV Comma Separated Values

CVC Content Verification Certificates

DENNuRL Dynamic Evolving Neural Network using Reinforcement Learning

DET Detection Error Trade-off

EANN Evolutionary Artificial Neural Network

EER Equal Error Rate

EPNet Evolutionary Programming Neural Network

EA Evolution approach

FEaR Feature Evaluation and Reduction

FNR False Negative Rate

FPR False Positive Rate

HTTP Hypertext Transfer Protocol

HEANN Hybrid Evolutionary Artificial Neural Network

MDP Markov Decision Process

MITM Man In The Middle

MSE Mean Square Error

NN Neural Network

NN-SAGP Self-Adaptive Growth-Brobability based Neural Network

OMNN Optimization Methodology for NN

PEDS Phishing Email Detection System

POMDP Partially Observable Markov Decision Process

PA Pruning approach

RL Reinforcement Learning

RL-Agent Reinforcement Learning Agent

ROC Receiver Operating Characteristic

SEP Square Error Percentage

SMTP Simple Mail Transfer Protocol

TD Temporal Difference

TER Testing Error Rate

TNR True Negative Rate

TPR True Positive Rate

VNP Variance Vullity Pruning

URL Uniform Resource Locator

1

Introduction

1.1 Introduction

Criminal activity that targets Internet users has an important effect on computer security, which is one of the fundamental computer science disciplines. Along with the evolution of the Internet and information technology applications, many kinds of attacks and security issues started to appear. At the beginning of the 1990s, the Internet was becoming increasingly popular and accessible across the world, and security threats also began to evolve. Over the years, online customers started to conduct a new kind of trade, called electronic commerce (E-commerce). By the end of 2001, E-commerce represented $700 billion in transactions (Woodley, 2012), and this increased every year. By 2000, the majority of business companies, including banks, in the United States and Western Europe had started to provide services for customers through an Internet website (Keivani et al., 2012). Due to the development of Internet architecture and online trade, online customers are required to enter their own sensitive data (such as bank account numbers) on some websites. These sensitive data have been targeted by attackers, and a specific kind of attack started to appear in the mid-1990s which is

called phishing. Phishing is a kind of online identity theft using socially engineered attacks, and it forms one of the central issues considered in this thesis.

According to the Anti-Phishing Working Group (APWG) (APWG, 2015), the total number of unique phishing attack reports collected by this organisation from January to September 2015 was 1,033,698, which is twice the number of reports for the same interval in 2014. The increase in registered phishing attacks is due to phishers employing more sophisticated new techniques to lure online customers.

Stealing a person's identity is one of the most popular cybercrime activities. According to the USA Federal Trade Commission in 2015 (Commission et al., 2015), identity theft ranks second in frequency at 16% of all customer complaints, and registered at the highest rank for the last 15th consecutive years. Phishing can formally be defined as "*a criminal mechanism employing both social engineering and technical subterfuge to steal consumers personal identity data and financial account credentials. Social-engineering schemes use spoofed e-mails purporting to be from legitimate businesses and agencies, designed to lead consumers to counterfeit websites that trick recipients into divulging financial data such as user names and passwords*" (APWG, 2015). A typical phishing attack, as described in Figure 1.1, starts by designing a fake website and then sending a fake email to an Internet user which tries to convince him to follow a fraudulent link to a fake website which looks exactly like a legitimate one. If the user follows the spoof link, the attacker starts to collect user's security information which will later be used for illegal purposes.

Phishing attacks that use new techniques are called zero-day attacks (Ammar, 2014). Various classification models are trained to detect phishing attacks using sets of previ-

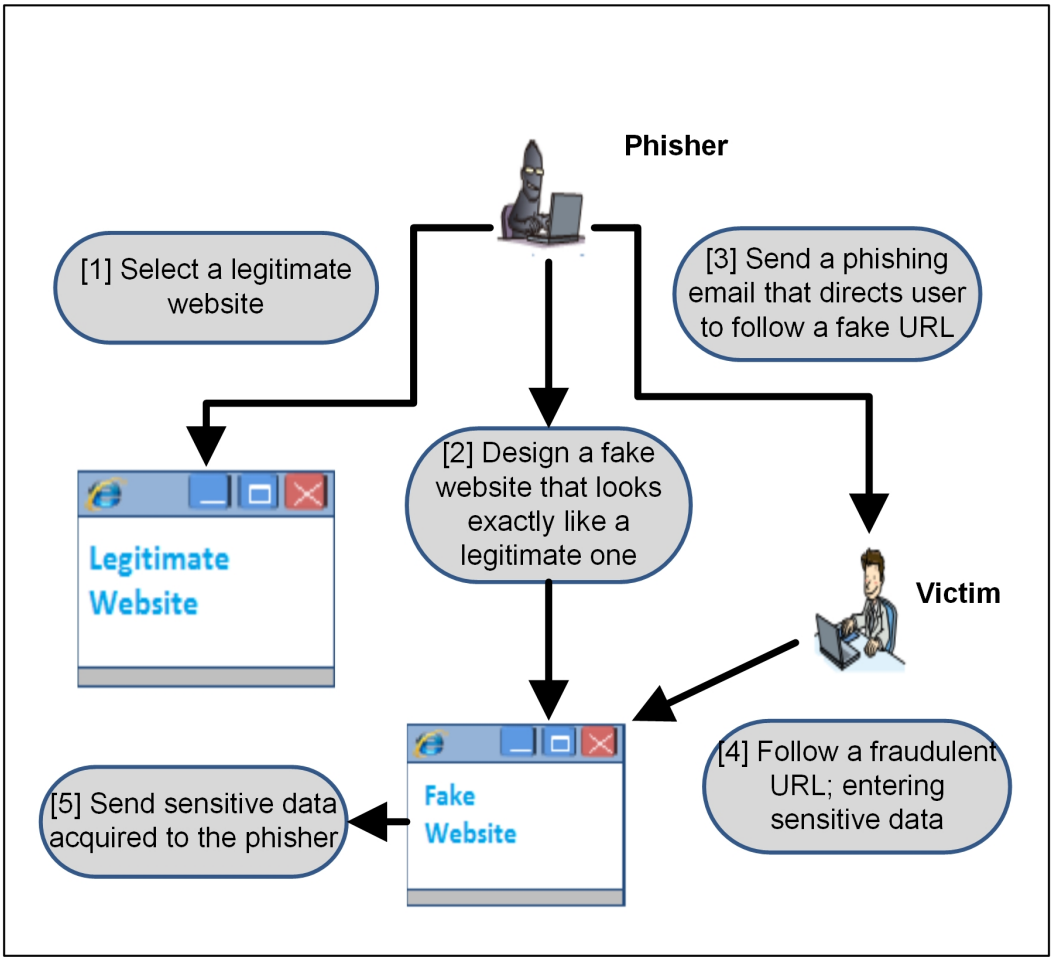


Figure 1.1: Typical phishing attack

ously known features but these cannot handle zero-day phishing attacks that use new behaviour. Techniques built using black-lists (Huang et al., 2009; Mohammad et al., 2015; Raffetseder et al., 2007) and white-lists (Han et al., 2012; Li et al., 2012) fail to detect new phishing attacks, due to the fact that the lifetime of a phishing website is on average 46 hours (Na et al., 2014), and since phishing attacks may occur before the web page has been added to the blacklisting.

Anti-phishing techniques based on a fixed number of features and trained using an offline dataset will also fail to detect new phishing attacks in the online mode, and classification errors will increase over time (Ahamid et al., 2013; Hu et al., 2014). The phisher tries to deceive online customers by using communication channels such as

email (Mohammad et al., 2014), SMS (smishing) (Kang et al., 2014), vishing (voice phishing) (Collier and Endler, 2013), and instant messaging (Luo et al., 2013). Phishing detection and prevention methods can be applied with any of these communication channels, or at web page level. According to Abdelhamid et al. (2014), 65% of phishing attacks use email as the main communication channel to lure online customers (Abdelhamid et al., 2014). The authors built a phishing detection and protection model to handle phishing attacks at email level. The detection of phishing at an earlier stage (at email level) of the attack will provide a more secure environment for Internet users, but if the detection happens at late stage like web page level the Internet user will already be in the middle of the attack, and some malware may have been installed on his device.

Researchers studying the phishing attack problems have proposed many solutions that detect attacks at many different levels. Some detect phishing attacks at web page level such as Aburrous et al. (Aburrous et al., 2009) and Barraclough et al. (Barraclough et al., 2013). Moreover, some models are built to detect phishing attacks at an earlier stage at email level, where the phisher is still trying to convince the user to go to the fraudulent web page (Khonji et al., 2012). This is a better strategy because, firstly, if the phishing attack is detected at webpage level, it will slow down navigation of websites by checking every Uniform Resource Locator (URL) the user try to open before grant access. Secondly, by detecting phishing attack at email level it is more secure for users since the attack is detected at earlier stage. For example, when the user opens a web page, some codes maybe downloaded on to the user's device to infect it. Moreover, phishing email datasets are available at any time, but the average lifetime

of a fraudulent webpage is about 46 hours (Na et al., 2014).

Many solutions have been proposed for phishing detection. These solutions usually suffer from high false positive rates, and their accuracy still needs to be enhanced. Some previous solutions depend on data mining algorithms that use a set of features (Barraclough et al., 2013; Hamid and Abawajy, 2011). Other solutions depend on black- or white-listings, which are not so effective since the lifetime of a phishing website is very short. Others use content-based approaches with a lexical URL (Zhang et al., 2007a).

1.2 Research Motivation

The losses caused by phishing can be divided into direct and indirect losses (Anderson et al., 2013; Jakobsson and Myers, 2006). Direct losses affect individuals as well as organizations due to successful attacks, whereas indirect damages include the costs imposed on society due to a certain type of attack either or not it is successful. Defence losses include all the money and effort involved in the prevention of phishing attacks, and costs to society are the sum of all losses from previous attacks.

Direct damage may include money stolen from users bank accounts, the time and effort required to change account details after detecting attacks, lost bandwidth due to phishing emails, and the inconvenience caused when a user needs to access his account, and cannot gain access. Indirect damage may include the online customer's loss of trust in E-commerce, the closure of some channels of communication with customers and their banks such as instant messaging and email, and the time and effort needed to clean PCs from when malware has been installed during attacks.

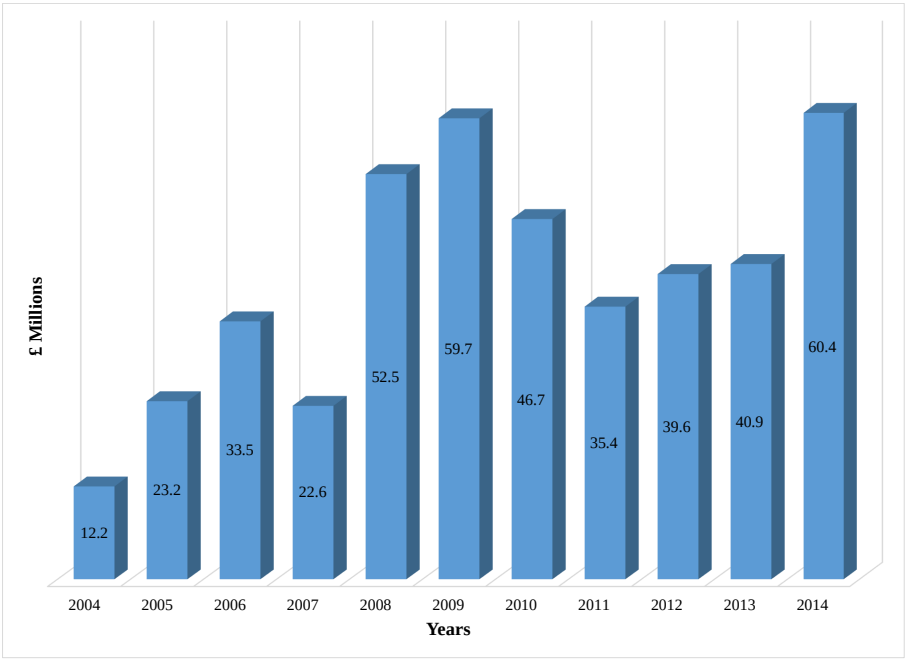


Figure 1.2: Online banking fraud in UK 2004-2014 (UK Cards, 2015)

Phishing has become a major problem in online transactions, accounting for £174.4 million lost in the UK in 2015 as reported by Financial Fraud Action UK (FFA UK), which was an increase of 23% compared to a loss of £142 million in 2013 (UK Cards, 2015). Changes in online banking fraud for the period from 2004-2014 is shown in Figure 1.2, where the highest registered loss occurred in 2014 with 47% greater loss than in the previous year (UK Cards, 2015). The increase in amount lost shows that phishing detection and protection techniques used until now have not solved the problem and do not properly protect online consumers. This is because phishers regularly change their techniques and strategies to lure online consumers to access their websites.

According to the APWG (APWG, 2015) the number of unique phishing emails re-

Table 1.1: Facts and financial damage related to phishing

Facts Related to Phishing			
	Cost	Period	Source
Number of unique phishing websites detected	630,494	1st-3rd quarter 2015	APWG
Number of unique phishing email reports	1,033,688	1st-3rd quarter 2015	APWG
Number of brands targeted by phishing campaigns	3,774	1st-3rd quarter 2015	APWG
Country hosting the most phishing websites	USA	1st-3rd quarter 2015	APWG
Number of identity thefts registered in USA	490,220	January-December 2015	Federal Trade Commission
Financial Damage			
UK	Âč 174.4 million	November 2014-October 2015	NFIB
Globally	$5.9 billion	2013	RSA
Every large organization (more than 10,000-emplyee)	$3.77 million	Every year	Ponemon Institute

ported during the period from January to September 2015 was 1,033,688, as shown in Table 1.1. According to the Ponemon Institute (Ponemon, 2015), every large organization that has more than 10,000 employees suffers from direct and indirect losses amounting to about $3.77 million every year. Indirect losses include the amount of money spent on security program to protect organizations from phishing attacks. The National Fraud Intelligence Bureau (NFIB) and Get Safe Online have found that phishing attacks cost UK consumers around £174.4 million in 2015 (UK Cards, 2015). For the interval from November 2014 until October 2015, 95556 new phishing frauds were registered, which is a 21% increase compared to the same period in the previous year. Moreover, 25% of victims were scammed using email or telephone calls. Phishing using email is the most frequently used rout to lure online customers representing 75% of all phishing scams. Also, phishing by phone calls accounts for 12% of reported attacks.

The Consumer Sentinel Network (Commission et al., 2015) has stated that the annual number of complaints about identity theft had reached 490,220, which ranks second at 16% of all complaints received by the FTC (Federal Trade Commission). The

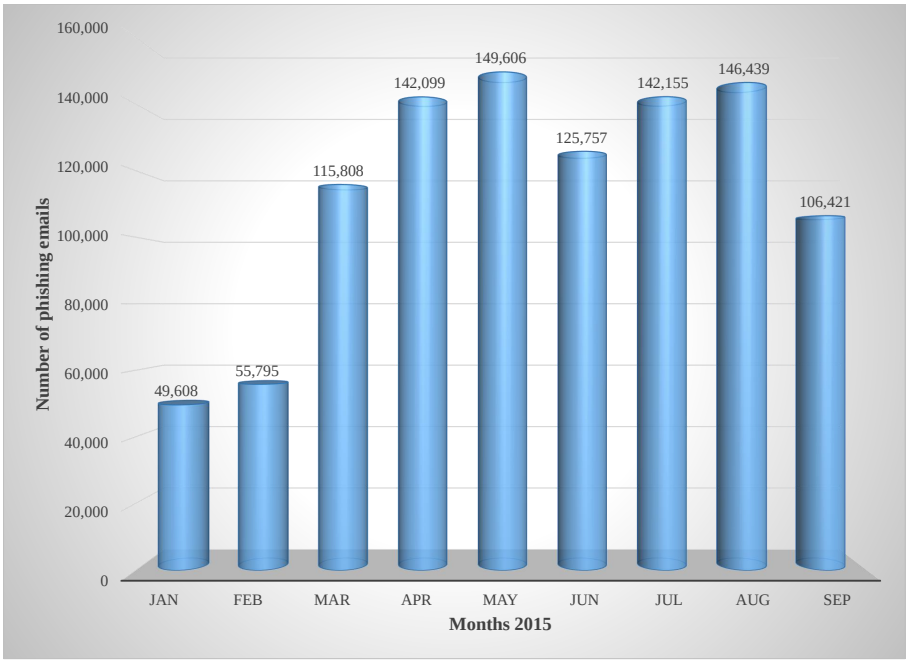


Figure 1.3: Number of unique phishing email reports submitted to APWG for the period from January to September 2015

global losses have been estimated by the RSA's fraud report to be $5.9 billion in 2013 (RSA Center, 2013). Phishing attacks target many industry sectors used by billions of people across the world. The number of phishing email reported shows an increasing number of phishing attacks over months in 2015 as shown in Figure 1.3.

From these estimates, it is clear that the phishing problem still adversely affects global online transactions. Furthermore, it may have a long-term effect if customers lose trust in E-commerce. The facts and statistics listed above lead to the intention to build an intelligent model that can detect phishing attacks and create a more secure environment for online customers.

1.3 Research Problem

Phishing problems affect the electronic commerce because online customers trust the Internet environment less (Kobayashi and Okada, 2013; Verma, 2013). Phishers use techniques which evolve to lure online customers, creating new phishing websites and spreading emails that try to convince Internet users to follow fraudulent links to access their websites. Phishing emails employ sophisticated techniques which direct the online customers to open a new web page, which has not yet been added to the black-list. A phishing attack that uses these new types of techniques is called a zero-day attack (Ammar, 2014).

Anti-phishing strategies can be divided into technical or non-technical solutions. Non-technical solutions used to protect the user from phishing attack depend on using awareness and training programmes to teach online consumers how to recognise phishing emails and websites (Alsharnouby et al., 2015). Technical solutions, however, depend on building detection and protection models based on training datasets. The non-technical solutions are very important, but they need to keep teaching Internet users about the new kinds of attack, and users need to read a lot of information. Furthermore, such training programmes may not be applicable for many kinds of users, such as older people and children. Moreover, this type of solution may be costly in the long term and it will not respond to new forms of attack immediately. Therefore, a solution is needed that gives protection to online consumers without their requiring intervention or action.

Any proposed model to handle zero-day phishing attacks where a new attacks are launched continuously, needs to have the ability to respond to the fact that the phishing

website life time is very limited (46 hours) where phishers launch a new attack to lure anti-phishing techniques designed based on black-list. Furthermore, the anti-phishing technique trained to detect phishing attack based on a fixed number of features will be lured by new attacks that use new techniques. Therefore, any anti-phishing technique that can response to zero-day phishing attack need to have the ability to reflect changes in the environment by making the feature selection a dynamic process.

In the proposed model, a neural network has been used because it gives the ability to modify the classifier to reflect changes in the environment so as to handle zero-day phishing attacks. Many issues must be dealt with to produce the best neural network that can represent a specific problem and how it can evolve to take account of new behaviour. To address these issues, this thesis proposes a new algorithm called the Dynamic Evolving Neural Network using Reinforcement Learning (DENNuRL), which produces a classifier that is modified automatically using a Reinforcement Learning Agent (RL-Agent). In order to solve the problems caused when a fixed number of features are used to build the protection model, a Feature Evaluation and Reduction (FEaR) algorithm is proposed.

1.4 Research Aims and Objectives

The main aim of this thesis is to develop a detection and protection model that will be able to discover and classify phishing emails in the online mode, with the ability to handle zero-day phishing attacks. This detection and protection model will be able to dynamically change to reflect changes in the environment, so that new phishing behaviours will be detected and utilized in updating the model. To achieve this main

goal, an adaptive neural network for the detection of phishing emails has been developed which uses reinforcement learning. Furthermore, the proposed model will have the ability to choose the best possible neural network.

To accomplish this aim, several intermediary objectives are listed below:

1. To conduct a literature review of previous studies in order to determine the gaps in existing knowledge. What are the current solutions? What are the main weaknesses?

2. To implement a pre-processing system which will have the ability to extract a set of features that represents all aspect of the phishing emails, and to determine its merits in the detection process.

3. To explore AI and data mining tools that could potentially be used in phishing email detection with high performance and accuracy.

4. To develop an online phishing email detection system using a dynamic evolving neural network based on reinforcement learning.

5. To show the accuracy and evaluate the performance of the proposed solution compared to other existing solutions.

6. To determine the ability of the proposed model in the detection of Zero-day phishing attack.

7. To evaluate the ability of the reinforcement learning approach used in the detection of phishing attacks.

8. To test the ability of an adaptive NN in the enhancement of the model generated while the system handles a new attack.

A quantitative research methodology has been developed and implemented to accomplish the aforesaid aims and objectives, including all experiments, data collection tools, and tests of the model and algorithms.

1.5 Research Contribution

Phishers regularly use new and sophisticated techniques to acquire sensitive information from online customers for illegal financial gain, and it is a challenging task to address this issue. Despite state-of-the-art solutions to this problem, there is still a lack of accuracy in online solutions, which cause loopholes in web-based transactions. These loopholes can be identified because the solutions used to defend against phishing attacks do not stop or reduce the losses caused by phishing, as shown in previously in Table 1.1. This research investigates these loopholes by developing a detection and protection framework using Artificial Neural Network (ANN) and reinforcement learning technology. An adaptive environment is developed that can detect zero-day phishing attacks and which will dynamically adapt itself to handle new phishing behaviour. This thesis therefore makes a set of novel contributions in the field of network security, as described in the following discussion.

1.5.1 Pre-processing algorithm

After investigating a massive number of phishing emails, handling previous phishing detection strategies, and analysing different phishing studies, technical reports, and research papers, fifty features have been extracted in this study.

A pre-processing system is proposed which uses the JAVA programming language, and this system is responsible for feature extraction from the main contents of emails. The fifty features were chosen from three sources: email headers and content and external sources. Many techniques and algorithms have been developed to process datasets in order to extract behaviour that distinguishes between phishing and ham emails.

To show the merits of the proposed pre-processing system, an offline detection model is proposed to compare between our pre-processing systems and other system. The high efficiency and low false positive rate registered are due to the pre-processing system as well as the features used. In the proposed model the same dataset and algorithms used to conduct the comparison. Furthermore, the proposed model is trained using a set of data mining algorithms used in the literature to determine the best one to be used in this field.

1.5.2 Feature Evaluation and Reduction algorithm

A new algorithm is proposed to handle the exploration of new phishing behaviours available in a selected dataset. This algorithm will be able to increase the performance of the proposed model, and the vast numbers of potential features will be reduced to the most effective number of features.

The number of important features will be changed according to changes in the dataset, which will allow the system to adapt dynamically to such changes. In addition, the selected most effective features will have an important role in reducing the complexity of the Neural Network (NN) used, which is the core of the proposed detection model, by reducing the number of hidden neurons and connections. Finally, an easy way to rank the selected features will be provided, and determine the rank for every feature.

1.5.3 Dynamic Evolving Neural Network using Reinforcement Learning

Selecting the best NN that can be used to solve the classification problem when classifying emails as phishing and ham emails has an important role in the classification process.

The characteristics of the NN include the number of layers, the numbers of neurons in each layer, and the appropriate number of training epochs. A novel algorithm is proposed in this study called the DENNuRL to address all of the mentioned above points in building the best neural network to solve the problem using the best possible NN architecture. Reinforcement Learning (RL) is used for the first time in this field to build a NN used to classify email as phishing or ham email. The proposed algorithm will have the ability to handle common problems encountered when NNs are used, such as overfitting and underfitting problems.

1.5.4 Phishing Email Detection System

In this thesis, a novel phishing email detection system is proposed that is dynamically adapted in the online mode. The proposed techniques are based on concepts of data mining, ANN, supervised machine learning, and RL combined to produce the Phishing Email Detection System (PEDS). The system generated is a NN will be adapted automatically without any user intervention to reflect changes and new behaviour in the up-to-date dataset.

The RL-Agent will control the adaptation process. The development of the PEDS includes building a new dataset, explore any new behaviours, and change the detection model which reflect these changes. RL has been used for the first time in this field.

1.6 Thesis Outline

In total, this thesis consists of six chapters, each of which concludes with a summary and conclusion intended to capture the essence of the material covered respective chapter and its subsections before moving on to the next one.

Chapter 2 Background Information and Literature Review: This chapter gives an introduction to phishing problems. Relevant terminology is defined and a detailed view given of the effect of phishing attacks and the losses related to phishing. Detailed descriptions are given of the anti-phishing strategies used by other researchers, concentrating on phishing email detection.

Chapter 3 Pre-processing and Feature Extraction: In this chapter two algorithms are proposed, which are pre-processing and FEaR algorithms. A selected list of features is determined and an algorithm is proposed to extract these features. Next the algorithm proposed to select the list of important features and explore new behaviour is discussed. Finally, an experiment is designed and carried out to show the merits of the proposed algorithm and the selected list of features.

Chapter 4 Dynamic Evolving Neural Network using Reinforcement Learning: The main algorithm responsible for generating the classification model is described in this chapter. An experiment is conducted to show the ability of the proposed algorithm to build a classification model, with a detailed comparison made with other strategies that use neural networks as the core of their detection models.

Chapter 5 Online Phishing Email Detection Framework: The final model used to generate the online phishing email detection system is described in this chapter. An

experiment has been designed to prove that the proposed algorithm can detect phishing emails with an acceptable error rate. This chapter ends with a comparison between the proposed framework and previous techniques used to classify emails into phishing and ham email.

Chapter 6 Conclusion and Future Work: The ïňĄnal chapter concludes this thesis with a summary of each chapter contents, findings, and provides a brief future outlook on how the proposed model can be enhance.

2

BACKGROUND INFORMATION AND LITERATURE REVIEW

2.1 Introduction

Phishing attacks are cybercrimes that affect many online transactions and are increasing in frequency. They are having an adverse effect on electronic commerce by leading online customers to lose their trust in online transactions. Phishing affects online transactions due to the insecure nature of most open networks that together constitute the World Wide Web, and, for example, Google bots detect 9500 new malicious webpages every day (Nguyen and Nguyen, 2016). Phishing attacks are difficult to trace and prevent and often go unnoticed. Sensitive information, once acquired, can be used without being detected by security systems and phishing leaves no records of intrusion into the targeted systems. To avoid such attacks, the user's data need to be protected from being stolen. The protection mechanism used can apply many strategies, such as removing phishing emails, warning and notification of inconsistent links and websites that could potentially harm the user or their computer by jeopardizing the integrity of their information. This chapter gives detailed insights into what a phishing attack is (Section 2.2) and reports on the history of phishing attacks (Section 2.3). A detailed discussion is then given of anti-phishing techniques in Section

2.4 which use non-technical and technical solutions. Then more detail is given of the technical solutions proposed by other researchers (Section 2.5). Section 2.6 discusses existing solutions that handle zero-day phishing attacks. Finally, Section 2.7 give a brief summery for this chapter.

2.2 Phishing - Definition, Lifecycle and Methods

The main intention of phishing attacks is to steal the internet user's sensitive data such as passwords, social security number (SSN) and credit card details. Phishing is a serious threat which causes billions of dollars of losses to business every year. Definitions and a taxonomy of phishing attack, lifecycle, and the main methods used are presented next.

2.2.1 Definitions related to phishing attacks

The terminology related to phishing has many different definitions in the scientific literature, ranging from merely descriptive to very broad, to scientific and general definitions. In previous studies, the phishing has not been clearly defined, and is sometimes described using examples (OpenDNS, 2016), while other authers seem to suppose that readers already know what phishing is, or that precise definitions would be too hard for readers to understand (APWG, 2015). Many authors have proposed their own definitions of phishing, leading to a large number of different definitions in the scientific literature. The most widely known definitions are presented below:

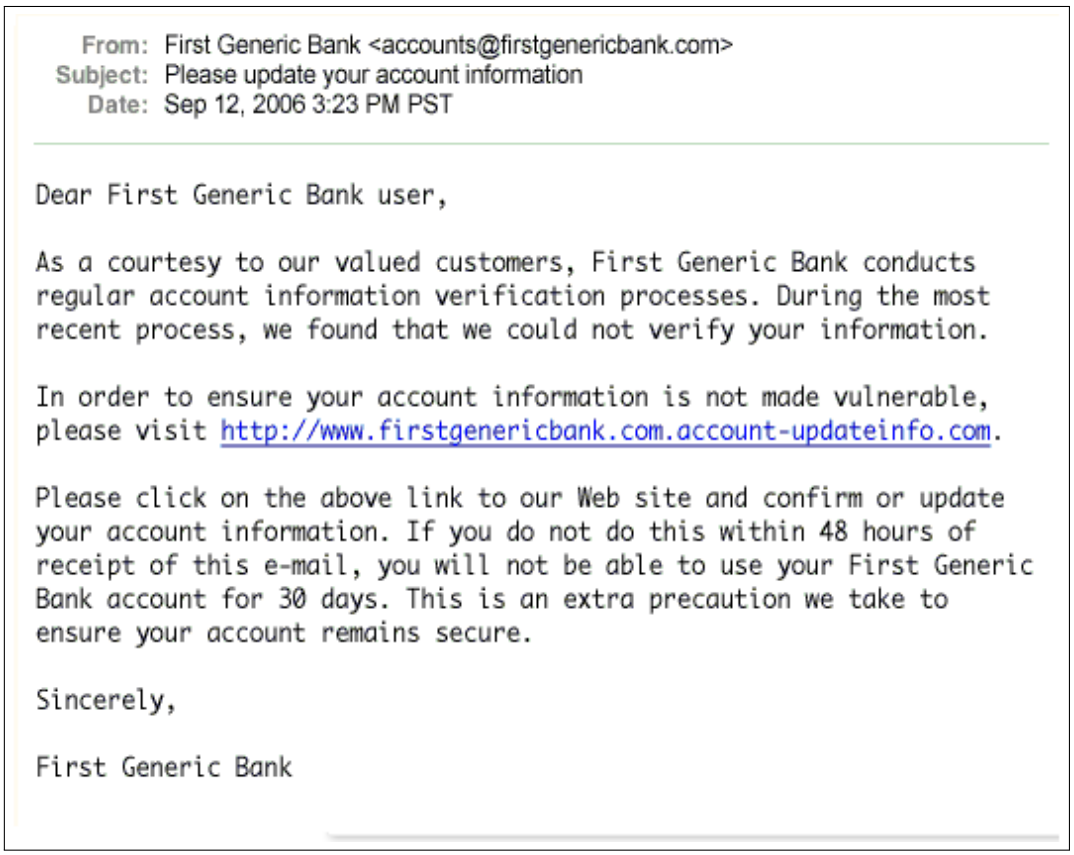
- The term phishing as defined by Oxford English dictionary is "*The fraudulent practice of sending emails purporting to be from reputable companies to induce*

individuals to reveal personal information, such as passwords and credit card numbers" (Aarts et al., 2014).

- PhishTank is one of the main collaborative clearing houses for data and information about phishing on the Internet. Its definition is that "*Phishing is a fraudulent attempt, usually made through email, to steal your personal information*" (OpenDNS, 2016).

- A longer definition has been proposed by the APWG, which is the most widely known organization unifying global responses to cybercrime across industry, government and law-enforcement sectors. There, phishing is a "*criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. Social-engineering schemes use spoofed emails purporting to be from legitimate businesses and agencies, designed to lead consumers to counterfeit websites that trick recipients into divulging financial data such as user-names and passwords*"(APWG, 2015).

The above definitions of the term phishing indicate that each organization has developed their own description of this term. Lastdrager (2014) tried to solve the problem of finding a standard definition of the term phishing by gathering 113 distinct definitions and then combining them into one definition using criminal science theory as a theoretical framework (Lastdrager, 2014). The resulting definition is that "*Phishing is a scalable act of deception whereby impersonation is used to obtain information from a target*". This definition can be considered to be the most authentic general definition of phishing, which can cover all kinds of phishing attacks. As shown in all of the previous definitions a typical phishing attack starts by sending an email to an online



From: First Generic Bank <accounts@firstgenericbank.com>
Subject: Please update your account information
Date: Sep 12, 2006 3:23 PM PST

Dear First Generic Bank user,

As a courtesy to our valued customers, First Generic Bank conducts regular account information verification processes. During the most recent process, we found that we could not verify your information.

In order to ensure your account information is not made vulnerable, please visit http://www.firstgenericbank.com.account-updateinfo.com.

Please click on the above link to our Web site and confirm or update your account information. If you do not do this within 48 hours of receipt of this e-mail, you will not be able to use your First Generic Bank account for 30 days. This is an extra precaution we take to ensure your account remains secure.

Sincerely,

First Generic Bank

Figure 2.1: Phishing email example (OpenDNS, 2016)

customer, as shown in Figure 1.1.

This is the beginning of the phishing attack, and it is where the approach proposed in this study tries to stop the attack and protect the Internet user from being deceived. An example of a phishing email is shown in Figure 2.1, and its main components are described in Figure 2.2. These examples and their annotations are collected from the PhishTank website (OpenDNS, 2016). The fraudulent link that the phisher inserts as a decoy in this email is for a fake website as shown in Figure 2.3. This fake website is designed by the attacker to convince the email recipient that it is a legitimate site by looking exactly like the original website. Figure âĂŐ2.4 shows where the email recipient is expected to enter his/her secret data, and if he does, the attacker will steal this information and may use it illegally later.

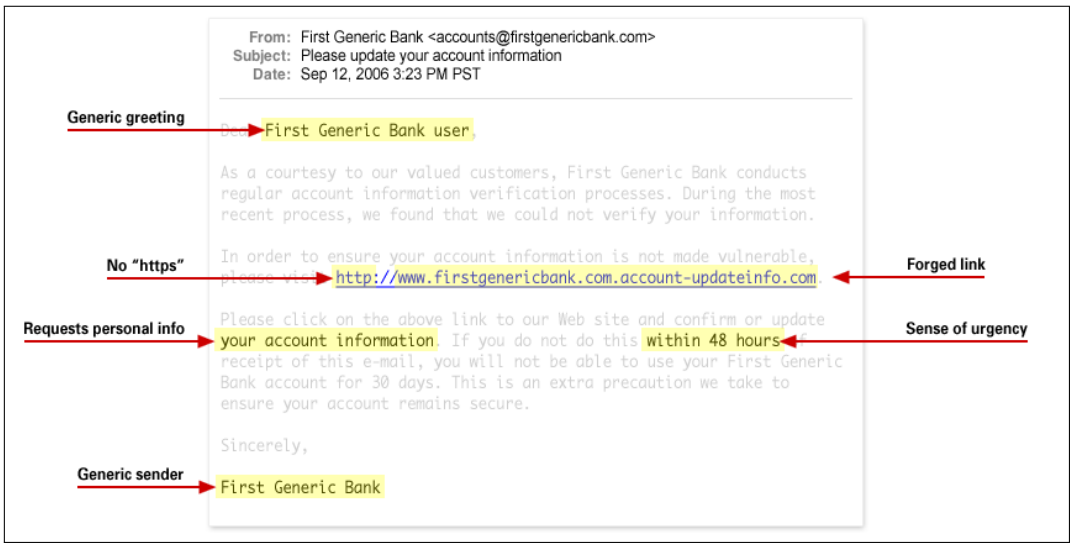


Figure 2.2: Phishing email example with annotations that describe its main parts (OpenDNS, 2016)

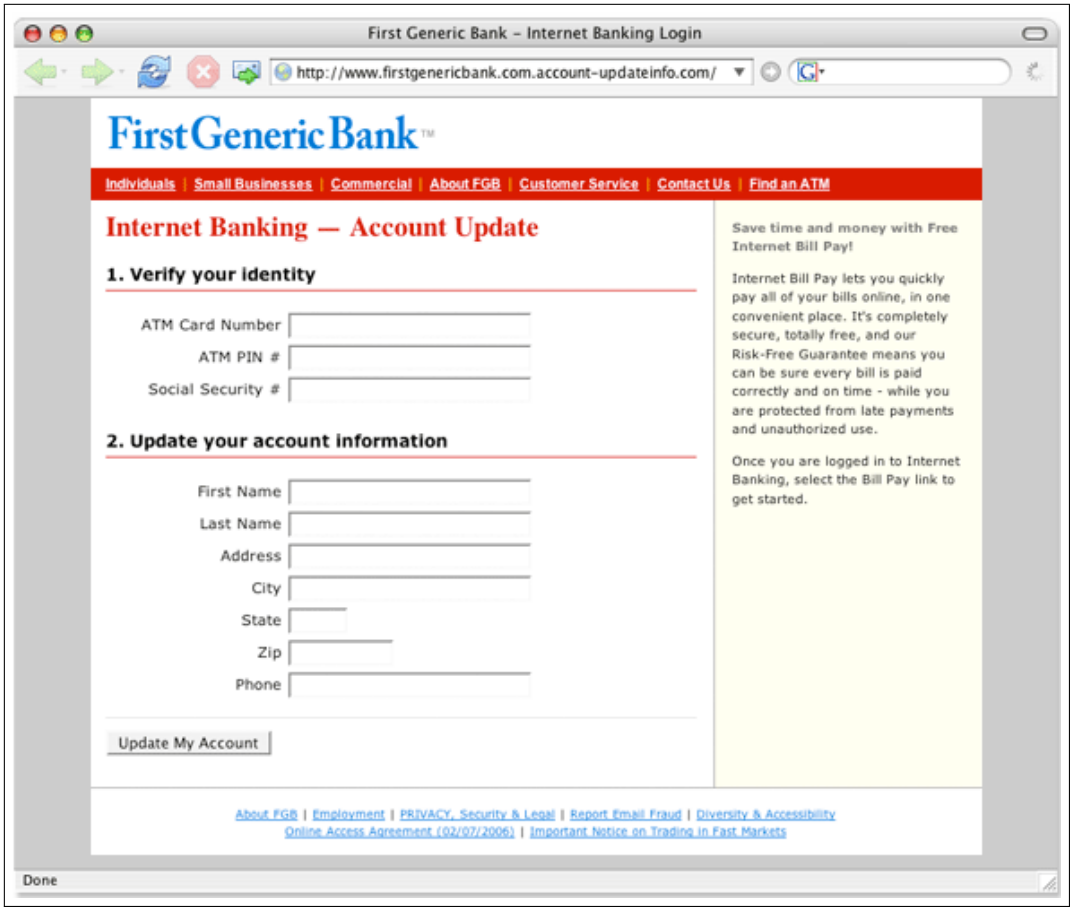


Figure 2.3: Phishing website (OpenDNS, 2016)

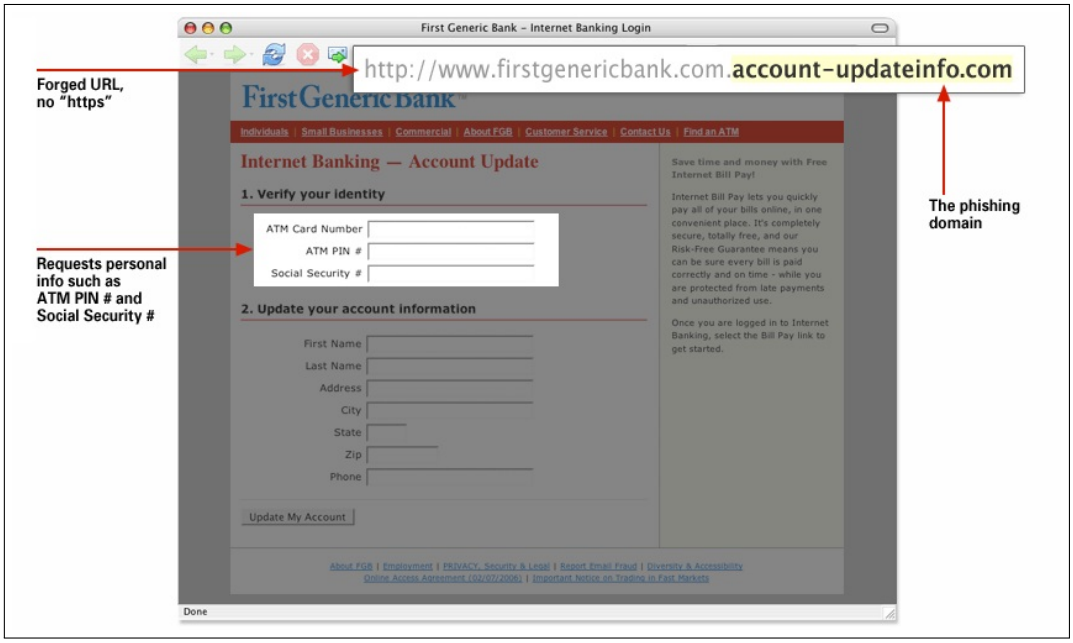


Figure 2.4: Phishing website with annotations (OpenDNS, 2016)

2.2.2 Phishing attack lifecycle

The lifecycle of a phishing attack closely resembles the fishing process, from where the word phishing originated. The fisher starts by preparing the bait that the fish will voluntarily catch, and the phisher begins in the same way by preparing the bait which is the phishing website. As the second step in the fishing process, the fisher presents a hook for the fish and waits until one of them voluntarily catches it. Similarly, in phishing, the phisher sends an email like that in Figure 2.1, to a large number of victims (Internet users). The email contains a hyperlink that points to the website designed in the first step and waits until one of the victims takes the bait and logs in to the phishing website and starts entering his data. Lastly, the phisher collects the victim's data (such as bank account information, Facebook account data, and email login information). Later, the phisher may use these data to steal money or information from victims.

2.2.3 Phishing attack methods

Phishers use many different techniques to initiate phishing attacks, the main methods used nowadays are email, SMS, Social network/media, VoIP, advertisements, instant messaging, search engines and malicious websites (Maurer, 2014). Phishers always modify their methods to use any communication method available to reach their victims. According to the APWG most phishers use emails to start their attacks (APWG, 2015). The same text used in phishing emails are also sent via other communication channels to reach online customers. In this thesis, a detection method is built to detect phishing attacks at the email level, since this is the most popular method used to spread these attacks.

2.3 History of Phishing Attacks

The term phishing started to appear after the initial period of the use of the Internet. Such frauds were initiated in 1995 but did not become widely known until nearly ten years later (Jakobsson and Myers, 2006). However, although many solutions have been proposed for this kind of attack, the losses caused by phishing still affect online transactions.

2.3.1 Origin of the word phishing

Phishing fraud uses spoofed emails and websites to deceive online consumers who voluntarily give sensitive information. As discussed previously, the term “phishing” comes from its similarities with the procedure of fishing. But there is also another reason for the use of “ph” in place of the “f”, Some of the earliest hackers commonly

replaced the letter f with ph in the previous form of crime called phone phreaking. Phreaking was first used by John Draper, aka Captain Crunch, who invented the infamous "Blue Box that emitted audible tones for hacking telephone systems in the early 1970s" (Rader and Rahman, 2015).

2.3.2 First fishing attack registered

The term "phishing" was used for the first time on January 2, 1996 (Zhang et al., 2012). The attack was registered in a Usenet newsgroup "alt.online-service.america-online". This first phishing attack targeted America Online customers and was the first form of what has become the most serious kind of attack that affects E-Commerce over the Internet. Moreover, this kind of attack has continued to grow and identity theft has ranked first among all types of cybercrime over the last 15 years according to the USA Federal Trade Commission (Commission et al., 2015).

2.3.3 First phishing attack description

The first phishing attack started by using algorithms to generate randomized credit card numbers, and then used these random numbers to open AOL accounts. The attackers used a program called AOHell to distribute the attack to AOL users (Collinson, 1995). The AOL company tried to put an end to this attack by constructing security procedures to prevent the use of randomly generated credit card numbers. The phishers sent instant messages and emails to the AOL users, acting as if they were an AOL employee. In the message that the AOL user received, the phisher asked the user to verify some secret information, such as an account number or any other private information. The problem got worse when attackers created AOL instant messenger

(AIM) accounts over the Internet which could not be traced by the AOL terms of service (TOS) (Reust, 2006). Finally, AOL tried to increase the awareness of its users, by sending email and instant messaging warnings not to provide sensitive information (Dhamija and Tygar, 2005).

2.3.4 The evolution of phishing

Phishers continue to use the same concepts that were employed in the first attack to lure online costumers. In 2001, attackers started targeting payment systems. Despite the fact that an attack targeting E-Gold in June 2001 was not successful, it established new targets for phishers (Almomani et al., 2013a). In 2003, phishers started using a new group of domains that targeted famous websites like eBay and PayPal. Spoofed emails targeting PayPal customers were sent, and fraudulent links were inserted in the email body that would take the email reader to the spoofed websites. When the user entered his/her information, the phisher would steal the costumer's data for use illegal later. By the following year, phishing attacks had registered tremendous success. These attacks cost online customers billions of dollars every year (Khonji et al., 2013). Popup windows were used to obtain sensitive information from online customers, but since that time many other sophisticated techniques have been used by phishers.

In 2006, phishers started to use the voice-over IP to initiate phishing attacks, and this new kind of attack is called Vishing (voice phishing) (Castiglione et al., 2009). In a vishing attack, phishers first set up a voice email system using a voice-over IP. Then the phisher uses an automatic dialler to call a long list of victims and play a recorded message, or the attack is started by sending an email that tells customers to call a phone number to update their data. When consumers respond, they hear a recorded

voice message that requests the customer to enter confidential information which the phisher can use later.

A new kind of phishing attack was launched in the late 2007 which still remains active, which is called spear phishing (Krombholz et al., 2015). This is a highly targeted phishing attack. Rather than sending phishing emails to anyone in the usual way, the phisher sends spoofed emails to consumers that appear to originate from somebody they know. For example, phishers can send emails pretending that an organization's manager is asking employees to update their data (like login information) or pretending that the email has come from a friend on social networks. The extent of spear phishing increased dramatically between 2009 to 2011, and it has been responsible for some high-profile corporate data breaches (Caputo et al., 2014). This kind of attack is successful because the phisher can collect information about consumers easily from the Internet by the basic mining of company websites. In summary, phishers continue to improve their tactics using new techniques, targeting speciiňĄc groups, and using alternative channels to spread their attacks. The present research focuses on detecting phishing attacks at email level, since most phishers use this channel to initiate their attacks.

2.4 Anti-Phishing Technology

As recently as 2007, the anti-phishing strategies implemented by businesses to protect individuals, and organization were fairly simple, but did not meet the expectations of users. To date, many techniques have been developed to fight phishing attacks, including legislation and technology developed to protect online customers. The solutions developed can be used by individuals as well as by organizations, and can be cate-

gorised as technical or non-technical. Non-technical anti-phishing solutions include awareness programmes that target the user to teach them how to identify phishing attacks and how to handle them. These solutions are outside of the scope of this thesis, but the next section briefly introduces this type of solution. The second type of solution includes technical anti-phishing solutions such as developing a security system that protects online customers without the need for their intervention.

2.4.1 Non-technical anti-phishing solutions

The first line of defence against phishing attacks is to teach people how to recognize phishing fraud and how they should respond to it. Learning among Internet users is vital to make them aware of the known techniques that phishers use to lure online costumers. Employees are just like customers targeted by phishers, but it is more dangerous if the phisher can lure an employee. The loss when a company is targeted can include direct financial loss, confidential customer data being stolen, or the theft of intellectual property such as private agreements or company plans, which may cause losses of millions of dollars. For that reason, employee awareness programmes must be a part of all organizational training.

Many organizations design awareness programmes that intended to teach internet users about phishing, such as eBay's spoof email tutorial, The Federal Trade Commission's "An E-Card for You" game (Commission et al., 2006), "Recognizing Phishing Scams and Fraudulent Emails" by Microsoft, and the National Consumer League's, Internet fraud tips on phishing. Kumaraguru et al. (2010) test some of these training programmes, and showed that training is effective in stopping phishing attacks if online customers read the training materials (Kumaraguru et al., 2010).

Kirlappos and Sasse (2012) indicate that education will not give users the protection they need (Kirlappos and Sasse, 2012). The effectiveness of a training system was evaluated by testing 515 volunteers, and after the training programme was run multiple times, 17.5% of the participants in the experiment were still being defrauded and entering their sensitive information in the simulated phishing websites.

Salem et al. (2010) proposed a combined approach using tools and human learning to avoid and detect phishing email (Salem et al., 2010). To address this problem, they firstly used a proactive solution by giving users a security awareness programme. Secondly, a reactive solution used an intelligent system to detect phishing attacks at email level that started by extracting six features from each email which was then each classified by using the fuzzy logic based expert system. The main weaknesses of the method were the number of features selected, which did not fully represent all characteristics of phishing emails. Moreover, this method is static because the rules are built manually.

Awareness programmes are important but not adequate to handle the phishing fraud problem, for the following reasons. Firstly, phishers regularly change the techniques they use, and so any training programme may not include the new technologies and strategies used by phishers. Secondly, the cost of those training programmes for both individuals and organizations is high, and they need to be repeated and updated frequently. Thirdly, the lifetime of a phishing website is very short, usually less than 2.58 days (Moore and Clayton, 2007) and the median lifetime is much smaller at about 20 hours. Therefore, any trainning programme could not follow the changes in attack strategies and the number of warning emails issued by the security department of a

company could be huge. Finally, these training programmes cannot be delivered to large numbers of costumers, for example, non-employees. Therefore, the best solution to phishing attacks should be an intelligent system which gives online customers the protection they need without requiring any user intervention. Any such system should have the ability to adapt itself dynamically to reflect the changes in phishing attack strategies.

2.4.2 Technical anti-phishing solutions

Technical anti-phishing solutions are tools used for the detection and prevention of, or response to, phishing attacks. These solutions have the advantage of automatically detecting a phishing attack without needing user intervention. Many of these tools are incorporated into anti-virus software and web browsers, others are standalone programs. In the next section the most widely used technical anti-phishing solutions are introduced along with a detailed discussion.

2.5 Anti-Phishing Detection Methods

Current phishing detection and prevention tools aim to protect online customers from phishing using various methods described in this section. However, despite state-of-the-art solutions to address this issue, online solutions still suffer from a lack of accuracy, which causes enormous losses every year in online transactions. Technical methods used for phishing detection and prevention can be further categorised according to the techniques used. Additional details of these methods, with their advantages and disadvantages, and various studies investigating them are discussed in the following sections. An anti-phishing tool can be categorized from the point of view of

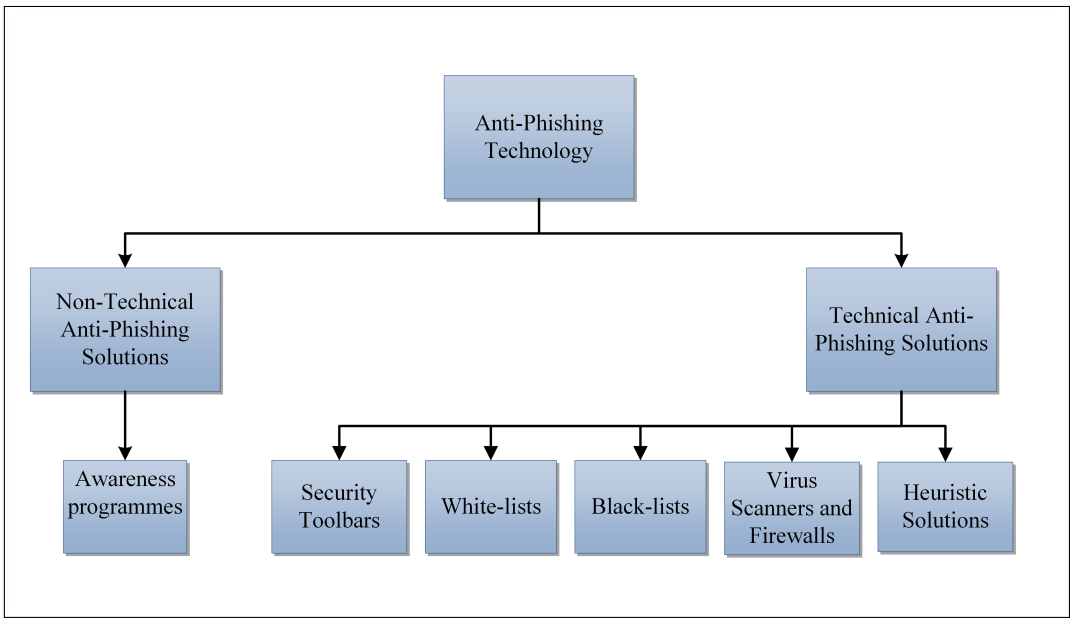


Figure 2.5: Anti-phishing technology

lifecycle into phishing detection tools, phishing prevention tools, or correction techniques (Khonji et al., 2013). Correction techniques are responsible for taking down a phishing attack; for example, in the case of the detection of a phishing website the cookies installed are removed. Prevention techniques are responsible for stopping a detected phishing attack in the future. Meanwhile detection tools are responsible for identifying phishing attacks, and in the following subsections detection methods are discussed in more detail. The most popular protection methods of blacklisting and whitelisting are considered first, and then additional security toolbars that can be installed in the web browser are discussed as shown in Figure 2.5. After that, the contribution of virus scanners in this field is discussed. Finally, heuristic-based solutions are considered, and then studies that try to solve zero-day phishing attack problems are covered.

2.5.1 Security toolbars

Security toolbars are a client-side tool embedded into a web browser to detect phishing attacks (Almomani et al., 2013a). Many toolbars have been developed to protect online customers from phishing, such as the Calling ID Toolbar, EarthLink Toolbar, Cloudmark AntiFraud Toolbar, eBay Toolbar and Netcraft Anti-Phishing Toolbar (Arachchilage and Love, 2014). Noman et al. (2013) examined a set of security toolbars attached to the most well-known web browsers to determine the best browser for the detection of phishing attacks, and it was shown that the best browser for protecting online customers was Google Chrome (Mazher et al., 2013). The main weakness with the security toolbar is that it is the user's responsibility to read the warning messages sent. Furthermore, the user must choose whether or not to proceed after the warning message has appeared, and Yen et al. (2011) found that 23% of those interviewed in their study did not read security toolbar warning messages, leading to spoof rates of 40% (Chen et al., 2011). The following list briefly describes the most widely used toolbars used to detect phishing attacks:

- PhishTank SiteChecker: a Mozilla Firefox add-on that blocks phishing websites based on the PhishTank database (DoleÅ¿al, 2008; Maggi, 2010).

- Netcraft toolbar: this depends on blacklist and whitelists maintained by Google, and it works by not allowing users to enter information at known phishing websites (Likarish et al., 2008).

- eBay toolbar: this protects eBay customers from being attacked by identifying a legitimate eBay website (eBay, 2016; Zhang et al., 2007b).

- EarthLink Toolbar: this toolbar retrieves visual notification for every website, which could be saved, neutral or negative (Abu-Nimeh and Nair, 2010; EARTHLINK, 2016).

- GeoTrust Trustwatcher toolbar: this server-based toolbar helps customers to identify trusted websites, and alerts user to unsafe websites (GeoTrust, 2016).

- SpoofGuard toolbar: this is a browser plug-in which watches customer activity and calculates a spoofed index. If the value calculated exceeds a threshold value, the spoofGuard warns the user about suspicious activity (Boneh et al., 2007; Gupta and Shukla, 2015; Kang and Lee, 2007).

- CallingID toolbar: this toolbar embedded in Microsoft Internet Explorer uses 54 verification tests to classify web pages as legitimate, low-risk or high risk (Gupta and Shukla, 2015).

- Cloudmark Anti-Fraud Toolbar: the evaluation of website legitimacy with this tool depends on user ratings. The final assessment is shown as a coulor icon; if the icon is green then it is legitimate, if red it is phishing website, and yellow a suspicious website (Gupta and Shukla, 2015).

- SpoofStick toolbar: this security toolbar displays the real domain for the website visited, where the phishers try to deceive online customers by using a domain that looks like the legitimate one. For example, if the visited URl is (www.ebay.com.ww21.us) the SpoofStick display the real domain as (ww21.us), which may decrease the opportunity to deceive online customers (Abu-Nimeh and Nair, 2008; Mao et al., 2013).

- TrustBar toolbar: this toolbar is based on the idea that a legitimate website uses a secure web connection SSL to send and receive confidential customer data. It displays a warning if the website does not use SSL to send or receive user data (Herzberg and Margulies, 2012).

- DOMAntiphish toolbar: this anti-phishing toolbar compares the Document Object Models (DOM) for the web page under investigation and a legitimate one. If there is no similarity, then the web page is considered to be an illegitimate one (Medvet et al., 2008; Nguyen et al., 2014).

- PwdHash: this is a browser plug-in that handles Man-In-The-Middle (MITM) attacks. It uses a hashing algorithm to merge data entered by the user, such as password for the website domain, which will make the data stolen by phishers meaningless (Blocki and Sridhar, 2016).

In the fight against phishing many different commercial applications has been developed. These solutions are developed as a standalone program (such as McAfee and Norton) or browser add-ons used to detect and block phishing attack (Kim et al., 2015; Ramzan, 2010). In 2009, Sheng et al., evaluate the major commercial applications to show their merits in providing zero-hour protection (Sheng et al., 2009). There are several anti-phishing blacklists, the most famous one is operated by Microsoft, Google, or PhishTank. The anti-phishing blacklist contains URLs manually detected as phishing. Microsoft's blacklists are embedded in Internet Explorer, and Google's blacklists are embedded in FireFox and Chrome browsers. where as PhishTank blacklists are free access lists used by researcher to build and test their solutions.

Sheng et al. (2009) evaluate the performance of several commercial applications to

detect phishing attacks. The assessments include the following applications:

- Microsoft Internet Explorer version 7 (7.0.5730.11), version 8 (8.0.6001. 18241).
- Firefox 2 (2.1.0.16), Mozilla Firefox 3 (3.0.1).
- Google Chrome (0.2.149.30).
- Netcraft toolbar (1.8.0).
- McAfee Siteadvisor (2.8.255 free version).
- And Symantec Norton 360 (13.3.5)).

The above tools depend on blacklists in their decision, except for the Internet Explorer 7 and Symantec, which use heuristic techniques in addition to blacklists to build their decision. There experiment shows that techniques based on blacklisting were ineffective in detecting zero-day phishing attacks. These tools detect less than 20% of attacks at time zero (when the attack was lunched), and after 12 hour from the beginning of the phishing attacks they can detect 47% to 83% of the attacks. The best result registered was for Symantec Norton, which is same result at time zero (20%). However, after one hour from lunching the phishing attacks it detects 73% of the phish which shows a faster enhancement (2-3 times) than other tools. Finally, this study showed that zero-hour protection offered by blacklists had a False Positive Rate (FPR) rate of 0% but a True Positive Rate (TPR) of less than 20%.

The authors in Sheng et al. (2009) recognised a gap between research and commercial applications in terms of TPR. Academic studies have focused on techniques that based on heuristics (machine learning techniques) which registered high TPR though

somewhat high FPR. The heuristic techniques are preferred at identifying phishing attacks that have not been seen before (zero-day attacks). On the other hand, commercial applications depend mostly on blacklists, which have average TPR but zero FPR. In addition, the blacklist techniques do not have the ability to adapt themselves to adequately handle zero-day phishing attacks. Moreover, a phisher can also exploit blacklist by automatically generated URLs.

2.5.2 Black- and white-lists

Techniques based on black- and whitelists are used to prevent phishing, using a database of both trusted (white-list) and phishing (black-list) websites. The databases can be saved and updated on the client's machine, or the lists are saved centrally on the server's computer.

2.5.3 White-lists

Whitelists are lists of trusted websites that an Internet user visits regularly. This technique allows the user to navigate only to a website which has been previously considered to be legitimate. This method is very efficient in handling zero-day phishing attacks, and it also produces zero false positive results. The main disadvantage of using whitelists is that it is hard to manage all the websites that users will navigate in the future. Basically, when a user chooses to open a legitimate website, and this website is not listed in the whitelist, the system will consider it to be a phishing website, which increases the false negative rate. this means that whitelists are not very popular. Li et al. (2012) proposed a phishing prevention tool called IEPlug, which is based on whitelist. IEPlug inspects the websites visited by the user, and if a phishing website

is detected it shows the certificate authority dialog and gives a warning to the user (Li et al., 2012). The main disadvantage of using this technique is that the white-lists needs to be maintained manually and the visual effects may disturb users.

Han et al. (2012) proposed an Automated-Individual White-list (AIWL), which is a phishing prevention tool based on the idea of whitelist. The AIWL uses a NaÃŕve Bayesian classifier to trace user login attempts (Han et al., 2012). If the Internet user regularly accomplishes a successful login for a specific website, the AIWL then asks the user to add that website to the white-list. The main disadvantage of this system is that the user has to take responsibility for considering a particular website to be legitimate, whereas all other websites are considered malicious.

In conclusion, a white-list method may be very useful when used along with other methods, such as blackl-ists and heuristic approaches. It can be used to speed up the other phishing detection methods, where websites known to be legitimate do not need to be tested.

2.5.4 Black-lists

In black-list phishing prevention approaches the requested URL is compared with a predefined phishing black-list. Blacklist phishing prevention is a very well-known technique used to handle phishing attacks. Most famous web browsers, like Google Chrome and Internet Explorer, use black-lists for phishing prevention. If the Internet user tries to visit a fraudulent website already saved in the black-list, the web browser denies access or warns the user about that website. Furthermore, the black-list has a very low false positive rate, which leads many users to prefer it compared to other methods. The main reasons for the wide use of black-lists are their low false positive

rate, as mentioned above and also they are very simple to design and implement (Huang et al., 2009; Raffetseder et al., 2007).

Sheng et al. (2009) evaluated the effectiveness of phishing black-lists in detecting phishing attacks (Sheng et al., 2009). A number of fresh phishes were used in the evaluation process, where a fresh phish is a phishing attack initiated within the last thirty minutes. This study showed that tools based on black-lists are ineffective in the prevention of zero-day phishing attacks, more than 80% of which were not detected.

The main disadvantage of using a black-list is that it will not contain all phishing websites at any given moment, because the lifetime of these websites are so short. In the time between the initiation of the attack and its detection, the phisher may have deceived many customers and created another fraudulent website (Mohammad et al., 2015).

2.5.5 Virus scanners and firewalls

Antivirus software has an important role in the battle against phishing attacks, and many organizations invest large sums of money in antivirus solutions to detect and prevent phishing. But this investment does not stop the growing losses across the world due to phishing. Evidence of the success of the attackers as stated by Parmar (2012), is that 19% of spear-phishing attempts result in successful attacks (Parmar, 2012).

When using firewalls, an additional layer of security is inserted which gives additional control over the network traffic. The use of firewall programs can protect Internet users from phishing attacks that use Trojans and key loggers. These kinds of attacks try to steal the user's sensitive data and send it over the Internet to the phisher. Personal

firewalls are used to prevent the sending of sensitive user data. There are many well-known firewalls, such as Zone Alarm (Thakur et al., 2015), and Microsoft Windows Firewall (Izhar et al., 2013; Thiyagarajan et al., 2010). In addition, FraudEliminator is standalone software that works as a firewall, protecting the online customers from opening a phishing website (Ramesh and Divya, 2015).

The main disadvantage of using firewall and virus scanner software, as stated by Colin (Tankard, 2011) that phishing threats are specially designed to overcome controls such as firewalls and virus scanners. Furthermore, the success of phishing attacks is clear evidence that the methods used are not effective in detecting such attacks.

2.5.6 Heuristic solutions

Heuristic methods check for one or more feature extracted from the communication channel used to initiate the attack, such as email, website, SMS, or Instance Messages (IMs). The heuristic-based detection methods use artificial intellegence to build a classification model based on a training dataset. Next, that model is used to classify the data received as legitimate or a phishing attack. It employs the features identified in the classification process, such as a sender domain or URLs embedded in the email, or JavaScript code. The complete list of features used in this thesis is discussed later in Chapter 3. Heuristic techniques can be executed on the server or client machines. Furthermore, they can be used as part of other applications such as browser toolbars, firewalls, or antivirus software.

Toolan et al. (2009) used many classification algorithms to build a classifier ensemble to classify email as phishing or legitimate (Toolan and Carthy, 2009). The proposed model used five features selected manually by the authors to represent emails. The

detection model was developed in two steps. Firstly, the authors evaluated six different classifications algorithms, including C5.0, NaÃŕve Bayes, SVM, Linear Regression and K-Nearest Neighbour. C5.0 was proven to have better accuracy and precision, and a classifier ensemble was then built which contained a set of the best algorithms. In the next step, an R-Boost technique was developed that contains two phases. The emails are first classified using the C5.0 algorithm, and those classified as legitimate in the previous step are then reclassified using another classifier from the chosen set. This study proved that the best algorithm combination is C5.0 in the first layer and the SVM algorithm in the second layer with an accuracy of 93.68%. The main weakness of this technique is that, the proposed model does not have the ability to adapt itself to reflect changes in the environment. Furthermore, the features used are too limited and do not represent all aspects of a phishing email. In addition, the feature set is static, and will not reflect changes in phishing attacks.

A ternary classification approach has been proposed to filter email, which distinguishes between three message types of ham, spam and phishing (Gansterer and Pölz, 2009). Thirty features were used, nine of which were newly introduced by the authors. Feature extraction and classification are implemented as a plug-in for the Apache James Server, whereas for feature ranking the ratio gain algorithm in Weka was used. Finally, the MSN search engine was used to extract online features. An overall classification accuracy of 97% was achieved, but the main weakness is that further enhancement in the pre-processing phase is needed to increase the overall metrics. Furthermore, the authors did not take into consideration important metrics like true and false positives.

A different approach for phishing email detection was proposed by Hamid Abawajy

2011. This uses the Bayes algorithm as a classification algorithm with hybrid features that combine content-based and behaviour-based features. The hybrid features are, firstly, those in the email header such as subject-based features, sender-based features and behaviour-based features (Hamid and Abawajy, 2011). Secondly, the body-based features include: URL-based, keyword-based, form-based and script-based information. An accuracy rate of 96% was achieved, with 4% false positive and false negative rates. The main weaknesses of this work were the high false positive rate and the lower accuracy achieved.

A multi-tier classification method was proposed for phishing email filtering by Islam and Abawajy (Islam and Abawajy, 2013). They also proposed an innovative method for extracting the features of phishing emails based on a weighting of message content and message header. A multiple classification algorithm is used including SVM, AdaBoost, and NaÃŕve Bayes. They are divided the email classification on three tiers, and they use 21 different features. Each email is classified in the first tier and reclassified in the next tier using another classification algorithm, so that if the email is correctly classified it will be passed on to the analyser and subsequently sent to the appropriate folder. Otherwise, it will be reclassified using the classification algorithm in tier three, and then sent to the right mailboxes folder. The proposed model achieved 97% accuracy, and the main weakness is that the authers did not take into consideration the computing overhead and performance issues of using multiple classifiers in multiple tiers. Furthermore, their model is fixed and cannot be automatically adapted to handle new phishing attack behaviour. Finally, the overall metrics also needed to be enhanced.

A further study tried to determine the right combination of features that would give higher accuracy and better performance by Olivo et al. (2013). The proposed technique could be used to optimize detection engines rather than to develop a new one. The study showed that using six features may give the same rate of false positives and true positives as when using eleven features, but the use of only six features is more efficient. The main weakness here was that this system was tested on only a small dataset of 450 phishing emails, which may not fully represent all kinds of such emails (Olivo et al., 2013).

Aburrous and Khelifi (2013) proposed a fuzzy logic and data mining algorithm to detect phishing websites (Aburrous and Khelifi, 2013). The proposed model depends on 27-features that stamp the fraudulent websites and it works as follows. Firstly, in fuzzification linguistic descriptors are assigned to a range of values for each characteristic of a phishing email. Secondly, rule generation uses a classification algorithm with a set of data mining tools implemented in Weka to learn the relationships among the selected different phishing features. Thirdly, aggregation of rule outputs all discovered rules are unified. Fourthly, in defuzzification the fuzzy output of a fuzzy inference system is transformed into crisp output. The final accuracy achieved was only 86%, which is the one of the main weaknesses of this work. The main disadvantage when using this technique is that the phishing attacks are only detected when the user tries to open the phishing website, which is a very late stage of the process. Furthermore, it is the user's responsibility to read the warning message which may indicate that the website is suspicious.

A robust server-side model to handle phishing email detection has been proposed

by Ramanathan Wechsler (2012), which is called phishGILLNET (Ramanathan and Wechsler, 2012). Natural language processing and machine learning techniques were used in multi-layered method to build the proposed model. PhishGILLNET was tested with 400,000 emails, 10% of which were phishing, 10% were ham and the rest were spam. The experiments were conducted using 10-fold cross-validation. The proposed model achieved 97.7% accuracy for phishGILLNET2 in the classification of emails as phishing, spam, and ham. The main drawback of the study is that URL processing was not taken into consideration to determine if a hyperlink is for a phishing or legitimate website. Furthermore, the detection mechanism was based on a list of topics which can be avoided easily by new phishing attacks, which means that this solution will fail to handle zero-day attacks. In addition, the phishing email dataset used was not a proven and publically available dataset, and instead the author selected a subset of spam emails and considered them to be phishing emails.

Barraclough et al. (2013) proposed a neuro-fuzzy model incorporating five inputs to detect phishing websites (Barraclough et al., 2013). These five inputs, which were introduced for the first time in this paper, are: legitimate website rule, user-behaviour profile, PhishTank, user-specific site and pop-up windows. Based on these five inputs, 280 features were selected. The proposed system extracts 280 features from each website and takes these as input to the fuzzy inference system to generate fuzzy rules and then use it to build neural networks that will detect phishing websites. The main weakness here is that the set of features is static, and so the system will not have the ability to detect new phishing attack that uses features not listed in the system. Furthermore, the detection process happens at a late stage at website level, where the

user of this system will be in the middle of an attack.

Miyamoto et al. (2009) evaluate the performance of different machine-learning algorithms in distinguishing between legitimate phishing websites (Miyamoto et al., 2008). They used nine machine learning techniques: AdaBoost, Bagging, SVM, Classification and Regression Trees, Logistic Regression, Random Forests, Neural Network, NaÃŕve Bayes and Bayesian Additive Regression Trees. these were tested using eight different features that best described the phishing websites. They use an F-measure, error rate, and AUC as performance metrics in the evaluation. The results showed that the highest F-measure was 0.8581 for AdaBoost, and the lowest error rate was 14.15% for AdaBoost and the highest AUC was 0.9342 for AdaBoost. The main weakness in this paper is that the technique works on the website to protect the user from phishing attacks, which is not a good choice because the lifetime of such a site is so short. Finally, the overall metrics also needed to be enhanced.

A different heterogeneous system has also been proposed by del Castillo et al. (2007) to classify email as legitimate or fraudulent (Del Castillo et al., 2007). This system uses three classification algorithms in three steps. A NaÃŕve Bayes classifier is used that works on the textual content of the email, and a rule-based classifier then processes the non-grammatical features of the email content. Finally, if the email is classified as suspicious, the system visits every hyperlink in the body of the email and analyses the responses in order to classify the websites as fraudulent or legitimate. The main weakness of this method is that in the processing of suspicious emails every hyperlink in the email need to be visited, but many of them may be from legitimate websites. This is excessively time-consuming and uses large bandwidth.

Kathirvalavakumar et al. (2015) have proposed a model that uses an elimination pruning neural network technique to classify email as phishing on legitimate (Kathirvalavakumar et al., 2015). The proposed model extracts 18 feature that represent different aspects of emails. To evaluate the proposed model, a dataset consisting of 4000 phishing and legitimate emails was used. The evaluation of the proposed model shows a very high accuracy rate of 99.9%. The main disadvantage of this study it did not take into consideration the generalisation problem, which is a common problem when using a neural network as the core of the detection problem.

Pandey and Ravi (2012) proposed a technique based on text and data mining to detect phishing emails (Pandey and Ravi, 2012). The model extracted 23 keywords from the email content using text mining. 2500 emails were used to train and test the proposed model, 1250 of which were phishing emails, and the rest legitimate. Seven classification algorithms have been used to build the classification model, where the highest accuracy registered was 97.6% for Genetic Programming (GP). The main disadvantages of this study are that, firstly, only a small dataset was used to evaluate the proposed model. Secondly, the features used took into consideration only the list of keywords extracted from the email content, which therefore omits important features of other parts of emails such as the email header and other characteristic of the email body. Furthermore, the keywords can then be avoided by phishers in new attacks which will not be detected by this model. Finally, the proposed model does not have the ability to adapt in the online mode to reflect changes in new attacks.

2.6 Zero-Day Phishing Attacks

Phishing email detection has been investigated in many studies; however, only a few models have been designed to handle zero-day phishing attacks. Techniques using blacklists and whitelists will fail to detect a new phishing attack, because the average phishing website lifetime is only 46 hours (Na et al., 2014), and a phishing attack may thus take place before a new fraudulent web page is added to the blacklist. Moreover, anti-phishing techniques that are based on a fixed number of features and trained using offline datasets will fail to detect new phishing attacks in the online mode, and classification errors will increase over time. The increase in classification error rates will happen because new phishing behaviour that phishers may use in zero-day phishing attacks will not be recognized as phishing behaviour by the detection model built using a static number of features.

Almomani et al. (2013) proposed a novel framework that classifies email in the online mode into phishing and ham email (Almomani et al., 2013b). The proposed model is called the phishing dynamic evolving neural fuzzy framework (PDENFF). Here, a total of 21 features were extracted from each email which are grouped into four groups: spam, body, URL and header features. The generated set of features is called the short vector. In the next step, the framework generates basic rules, and then a dynamic evolving neural fuzzy inference system (DENFIS) is used to produce the fuzzy rules, which the system is able to add, delete or update in the online mode. Two datasets were used for training and assessing the proposed model. In the first dataset, 8000 phishing and ham emails were collected from Monkey.org (Nazario, 2015) and the SpamAssasin (Mason, 2005) corpora. In the second dataset, 2300 emails were collected from a mail

server at the NAV6 centre at the University of Sains, Malaysia, which contain 300 of phishing, and 2000 ham emails. An experiment was conducted to show the merits of the proposed model, and the first two experiments demonstrated the benefits of using the short and long vectors and compared the output of the PDENFF with other classification models for the offline dataset. The third experiment was applied to zero-day phishing attacks and the model tested with a third dataset achieved a performance level of 98%. The main weaknesses of the proposed model were, firstly, that the dataset used for assessing the model in the online mode was very small, consisting of 300 emails. Secondly, the authors did not show how the system evolved in the online mode.

2.7 Summary

This literature review has shown that phishing is a challenging type of attack that is hard to defend against, even with the highest security solutions, since it targets human weakness to attain its goal. Phishing is becoming more complex along with developments in communication channels, and many different phishing types are appearing. This suggests a wide range of research projects on social engineering and phishing is possible. However, to keep the research within manageable proportions, this thesis focuses only on detecting phishing attacks at email level. This chapter has identified previous research on the vulnerability to phishing threats and discussed the technical and non-technical anti-phishing methods developed to defend against phishing attacks, with a particular emphasis on heuristic-based solutions.

The techniques proposed to detect phishing attacks at webpage level are not satisfactory. If detection occurs at webpage level, the online customer will be in the middle

of an attack. If he ignores the warning messages, which is often the case, the phishing attack will be considered to be successful. Therefore, the detection of phishing attacks at email level will be more secure for online customers. In this thesis, the proposed model detects phishing attacks at email level.

From the literature, the most promising solutions classify emails based on the selection of hybrid features which combine features from the email content and headers. Therefore, this study is based on hybrid features that cover all aspects of emails, as shown in Chapter 3 where the selected list of features is described. The review has shown that the approaches proposed so far suffer from high false positive rates and low accuracy. Furthermore, some studies did not use approved datasets or only analysed a small number of instances which may not fully represent all kinds of phishing attack. Finally, there is a limited number of studies which tackle zero-day phishing attacks in the online mode, and in this solution a clear idea is needed of how to explore new phishing behaviour that may used by phishers. Therefore, this study proposes a phishing email detection system that handles zero-day phishing attacks in the online mode.

The next chapter describes the pre-processing proposed to extract the selected list of features that represent all parts of emails. In addition, it describes the proposed algorithm used to explore new behaviour.

3

PRE-PROCESSING AND FEATURE EXTRACTION

3.1 Introduction

This chapter focuses on the following key questions: what are the important features that can be used to classify a given email into phishing or ham email? What are the main components of emails? And how can we design feature selection to be a dynamic process? To answer these questions, a pre-processing system is designed that is capable

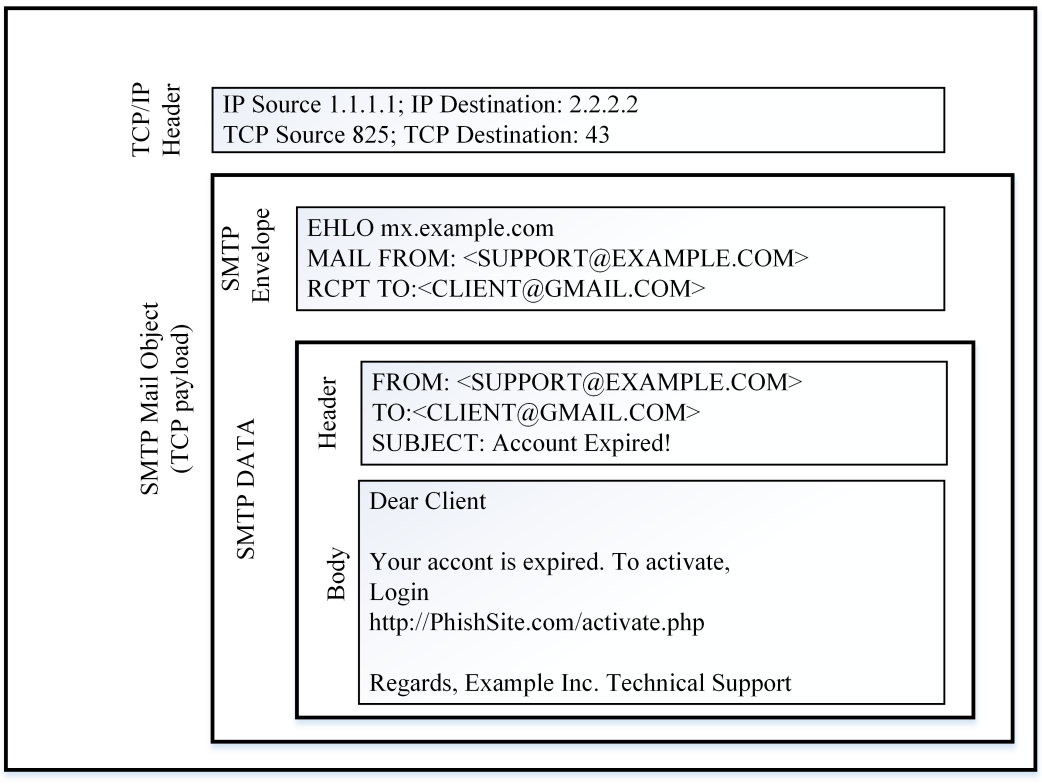


Figure 3.1: Email main components as described by (Khonji et al., 2012)

of retrieving a set of features from different email components. The extracted features are evaluated to identify the most effective list that can be used to classify the emails in training dataset. An algorithm is developed for this purpose which is called the FEaR algorithm.

A set of experiments was conducted to measure the performance of the pre-processing phase in phishing email detection. In addition, ten classification algorithms were used to build the classification model, and a comparison was performed to decide which of them are better to solve such a problem. The ten classification algorithms are selected from the literature as the mostly wide used algorithms used to build classification algorithms in this field.

3.2 Main Component of Emails

In 1978 VA Shiva Ayyadurai developed email for the first time (Aamoth, 2011). Since then, a lot of changes and enhancements have been made to the structure and content of emails to support new forms of usage. To make the exchange of emails between Internet users an easier process, the Advance Research Projects Agency (ARPA) developed the first standard RFC822 in 1982, which defined the main email components. Today's email format is an extension of the RFC822 standard, where the most recent email format is defined as the RFC5322 (Resnick, 2008). The email formatted using RFC5322 contain two main parts, the TCP/IP header and the Simple Mail Transfer Protocol (SMTP) mail object, as shown in Figure 3.1.

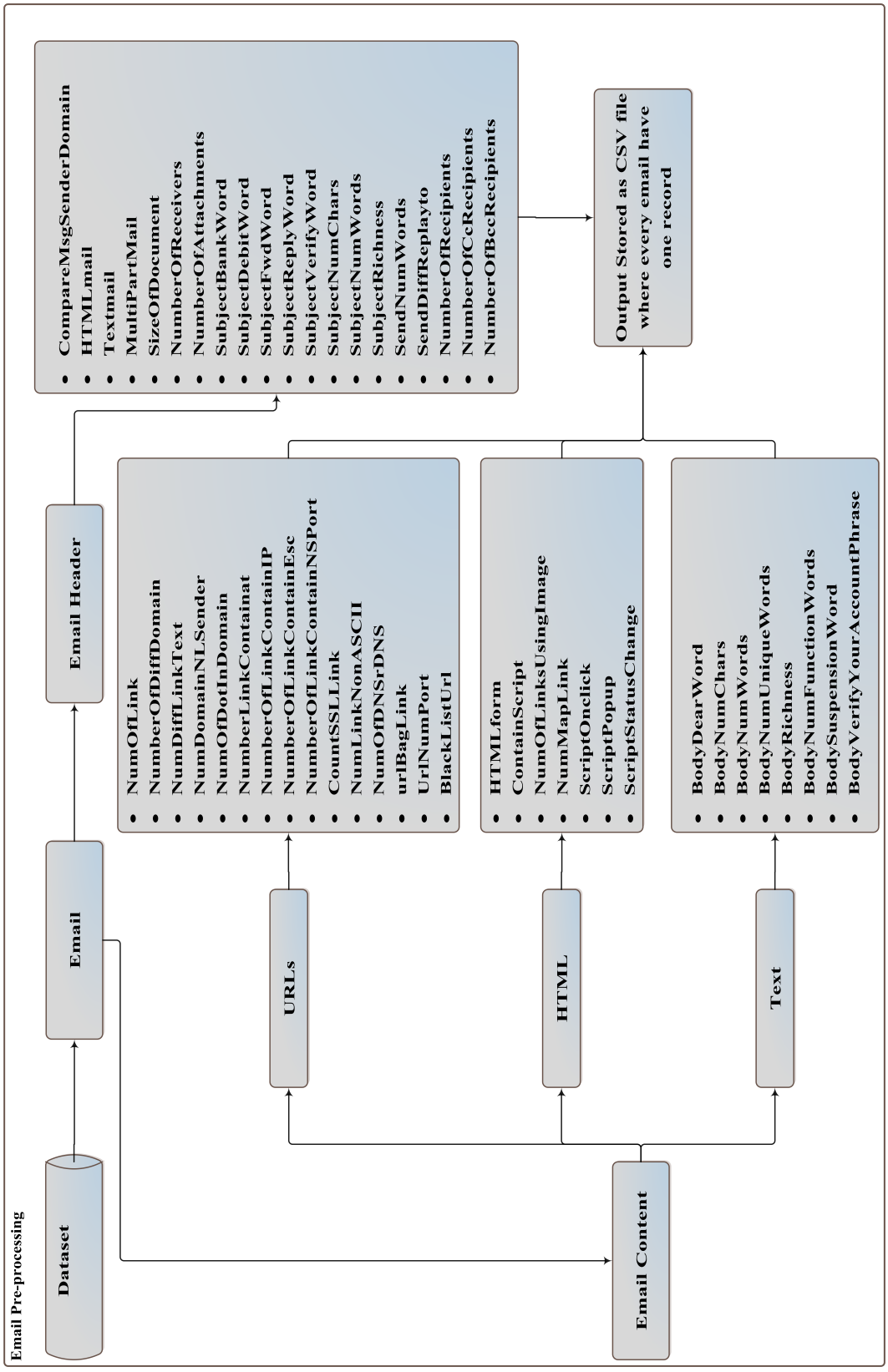


Figure 3.2: Main components of parts of the email

The SMTP mail object contains the SMTP envelope and data. The data part will contain the email header and, optionally, the email body. Once an email is received, the header is read to get the various details of the components of the message. Typically, the header fields of the email will contain information such as the sender, recipient, date, subject, and so on. In the next section a pre-processing system is described which will process the different email parts to retrieve information, which is then used later in the detection process.

```
Return-Path: <Bailey.ThereseWV9718@ezysurf.com>
X-Original-To: username@domain.com
Delivered-To: username@domain.com
Received: from spam.thegeekempire.net (dsl093-012-142.cle1.dsl.speakeasy.net [66.93.12.142])
        by spanky.domain.com (Postfix) with ESMTP id AF8FE538DC8
        for <username@domain.com>; Thu,  2 Dec 2004 23:10:34 -0500 (EST)
Received: from VERA-ATRIUM ([62.5.166.226])
        by spam.thegeekempire.net (8.10.1/8.9.3) with SMTP id iB34YE907726
        for <username@thegeekempire.net>; Thu, 2 Dec 2004 23:34:23 -0500 (EST)
Received: from 154.200.66.102 by 62.5.166.226 Fri, 03 Dec 2004 06:05:23 +0200
Message-ID: <qripcprjWUghpY4@rogers.com>
From: "beguiling Leann" <Bailey.ThereseWV9718@ezysurf.com>
Reply-To: "beguiling Leann" <Bailey.ThereseWV9718@ezysurf.com>
To: username@thegeekempire.net
Subject: eBay Bid Confirmed: Item Number 2372331171
Date: Fri, 03 Dec 2004 07:04:23 +0300
X-Mailer: Microsoft Outlook Express 5.00.2919.6700 bouffant awag
cacogastric-bedull: buscarle carposporangium augustness
MIME-Version: 1.0
Content-Type: multipart/alternative;
        boundary="--17634019336154049"
Status: RO
X-Status:
X-Keywords:
X-UID: 23

[               Priority: Normal               ]

----17634019336154049
Content-Type: text/html;
Content-Encoding: Bit21
```

Figure 3.3: Real example of email header

3.3 Pre-processing

The pre-processing phase contains two steps. The first step selects features to be extracted from each email text and header; these features describe the different properties of each email. The second step, includes selecting the most effective features from the set extracted in the first step. This will reduce the number of feature used in the proposed model, which will speed up the design and adaptation of the classification model.

The features are selected from three sources: email headers, email content, and external sources. A total of fifty features were selected, but the effect of each feature when classifying email into phishing or ham is not static. During the development of the system, the status of a specific feature could change dynamically to reflect the nature of zero-day phishing attacks. Four groups of proposed features are extracted from the different parts of the email: the email headers, URLs, HTML, and text as shown in Figure 3.2. To accomplish the pre-processing stage, the email is divided into header and content. From the content, the URLs, HTML, and text are extracted depending on the email content type. The email is divided into these parts to reduce the duration of pre-processing. In the following four sections brief descriptions of the features selected to represent phishing and ham email are given. The complete list of features, including those proposed for the first time, are shown in tables 3.1, 3.2, 3.3, and 3.4, where every feature is identified by an ID number to simplify the reference to each feature in later sections in this chapter. The newly proposed features with their IDs are: Textmail (3), MultiPartMail (4), NumberOfReceivers (22), NumberOfAttachments (23), NumberOfRecipients (47), NumberOfCcRecipients (48),

NumberOfBccRecipients (49), and BlackListURL (50).

3.3.1 Email header

Nineteen features were extracted from the email header, a full email header with all fields and their values is shown in Figure 3.3. Table 3.1 shows each feature and its ID for reference in this thesis. Full descriptions of the features extracted from the email header are shown in Table 3.1.

Table 3.1: Features extracted from email header

ID	Feature	Description
1	CompareMsgSenderDomain	This compares the message ID domain and sender domain and is a binary feature that determines if the domain names taken from the email sender are equal to those taken from the message-ID Toolan and Carthy (2010).
2	HTMLmail	Binary feature that determines if the email content type is TEXT/HTML Toolan and Carthy (2010).
3	Textmail (New)	This validates if the email content type is text/plain in the email header.
4	MultiPartMail (New)	A binary feature that validates if the email content type is multipart Smadi et al. (2015)
22	NumberOfReceivers (New)	Counts how many receivers are included in the (from) attribute in the email header.
23	NumberOfAttachments (New)	Counts how many attachments there are.
32	SubjectBankWord	Checks if the email subject contains the word (bank) and returns (true) if it exists and (false) otherwise Khonji et al. (2012).
33	SubjectDebitWord	Checks if the email subject contains the word “debit” and returns (true) if it exists and (false) otherwise Khonji et al. (2012).
34	SubjectFwdWord	Checks if the email subject contains the word (Fwd:) and returns (true) if it exists and (false) otherwise Toolan and Carthy (2010).
35	SubjectReplyWord	Checks if the email subject contains the word “Re:” and returns (true) if it exists and (false) otherwise Toolan and Carthy (2010).
36	SubjectVerifyWord	Checks if the email subject contains the word (verify) and returns (true) if it exists and (false) otherwise Khonji et al. (2012).
37	SubjectNumChars	Counts the number of characters the subject field contains and returns that number Toolan and Carthy (2010).
38	SubjectNumWords	Counts the number of words the subject field contains and returns that number Toolan and Carthy (2010).
39	SubjectRichness	Calculates the division between the number of words over the number of characters found in the subject field Chandrasekaran et al. (2006).
40	SendNumWords	Counts the number of words the sender field contains and returns that number c.
41	SendDiffReplayto	Checks if the email sender is not the same as the “Replay to” field, and returns (true) if they are equal and (false) otherwise Toolan and Carthy (2010).
47	NumberOfRecipients (New)	Counts how many receivers are included in the (To:) attribute in the email header.
48	NumberOfCcRecipients (New)	Counts how many receivers are included in the (Cc:) attribute in the email header.
49	NumberOfBccRecipients (New)	Counts how many receivers are included in the (Bcc:) attribute in the email header.

3.3.2 URLs

Table 3.2 shows the list of features related to the hyperlinks available in the email body. After extract all hyperlinks (URLs) from email content apply each of these features to the list of URLs. Table 3.2 shows full descriptions of these features.

Table 3.2: Feature extracted from URLs available in email content

ID	Feature Name	Description
6	NumOfLink	Computes the number of hyperlinks which appear in the email body Fette et al. (2007).
7	NumberOfDiffDomain	Counts the number of different domains that are used in the email content as hyperlinks Fette et al. (2007).
8	NumDiffLinkText	Computes the number of hyperlinks that have hyperlink text which does not contain the domain name of the hyperlink target Fette et al. (2007).
9	NumDomainNLSender	Computes how many hyperlinks use a domain which is not equal to the sender domain Fette et al. (2007).
10	NumOfDotInDomain	Computes the number of dots used in each hyperlink and returns the maximum number Fette et al. (2007).
11	NumberLinkContain@	Counts the number of links in the email body which contain the "@" character (Gansterer & Pölz, 2009).
12	NumberOfLinkContainIP	Counts the number of URLs in the email which contain an IP address Fette et al. (2007).
13	NumberOfLinkContainEsc	Counts the number of URLs in the email body which contain hexadecimal numbers or URL-escaped characters (Drake et al., 2004).
14	NumberOfLinkContainNSPort	Counts the number of URLs in the email body which contain a non-standard port (other than 80 or 443) Drake et al. (2004).
42	urlBagLink	A binary feature that returns (true) if any of the following words is found in the URL; these words are click, here, login, and update Bergholz et al. (2010).
43	UrlNumPort	Counts the number of URLs that contain a port in the authority section of that URL and returns that number Toolan and Carthy (2010).
50	BlackListURL (New)	A binary feature that returns (true) if there is any of the URL which exist in the email body exist in the black-list of URLs. These blacklist URLs are collected from PhishTank, which is a free community site where anyone can submit, verify, track and share phishing data. This feature is updated every 60 minutes to ensure that it contains the most recently registered phishing websites.

3.3.3 HTML

If the email content type is HTML, the preprocessing algorithm will extract the features described in Table 3.3.

Table 3.3: Features extracted from email HTML content

ID	Feature name	Description
5	HTMLform	Checks if the email content contains an HTML form element Bergholz et al. (2010).
15	ContainScript	Checks if the email contains a JavaScript pop-up Bergholz et al. (2010).
16	CountSSLLink	Counts the number of URLs in the email body which point to a website that encrypts the connection with a self-signed certificate.
17	NumOfLinksUsingImage	Computes the number of pictures which are used as hyperlinks Gansterer and Pölz (2009).
18	NumMapLink	Computes the number of pictures with image maps that are used as hyperlinks Gansterer and Pölz (2009).
19	NumLinkNonASCII	Counts the number of URLs that contain non-standard ASCII characters Gansterer and Pölz (2009).
21	NumOfDNSrDNS	Checks if the domain names have a corresponding reverse DNS entry and return (true) if they are equal, otherwise returns (false) Inomata et al. (2005).
44	ScriptOnclick	A binary feature that checks if an email contains onClick JavaScript event and returns (true) if it is available and return (false) if not Khonji et al. (2012).
45	ScriptPopup	A binary feature that checks if an email contains a JavaScript pop-up windows in its content and return true if it is available and false otherwise Khonji et al. (2012).
46	ScriptStatusChange	A binary feature that checks if an email contains a JavaScript that changes the text which appears in the status bar, and returns (true) if it is available otherwise returns (false) Gansterer and Pölz (2009).

3.3.4 Text

Extract the text shown to email reader from the email content and apply the features shown in Table 3.4.

3.4 Feature Extraction

The pre-processing phase is the most important phase of the proposed model because it handles the extraction of information from the dataset. A JAVA program was built to conduct automatic information extraction by reading the email. Pre-processing is divided into three different phases.

To accomplish the pre-processing task, the email is divided into two basic parts, the email header and body. The header contains information such as the message sender, receiver, message-ID, and content type. The message content contains the substantive

Table 3.4: Feature extracted from the email main text

ID	Feature name	Description
20	SizeOfDocument	Returns the email message size in bytes Gansterer and Pölz (2009).
24	BodyDearWord	Binary feature which checks if the email body contains the word "Dear" and returns (true) if it exists, or otherwise returns (false) (Khonji et al., 2012).
25	BodyNumChars	Counts how many characters are in the email body Chandrasekaran et al. (2006).
26	BodyNumWords	Counts how many words are in the email body Chandrasekaran et al. (2006).
27	BodyNumUniqueWords	Counts the number of unique words in the email body Chandrasekaran et al. (2006).
28	BodyRichness	Calculates the number of words divided by the number of characters found in the email body (Chandrasekaran et al., 2006).
29	BodyNumFunctionWords	Counts the number of function words discovered in the email body. Function words are account, access, bank, credit, click, identify, inconvenience, information, limited, log, minutes, password, recently, risk, social, security, service and suspended Chandrasekaran et al. (2006).
30	BodySuspensionWord	Checks if the email body contain the word "suspension" and returns (true) if it exists and (false) otherwise Khonji et al. (2012).
31	BodyVerifyYourAccountPhrase	Binary feature that checks if the email body contains the sentence "verify your account" and return (true) if it exists and (false) otherwise Khonji et al. (2012).

information intended for those who read the message. The content type depends on the message content-type attribute in the message header. The analysis in the present study considers the list of attributes shown in Figure 3.2.

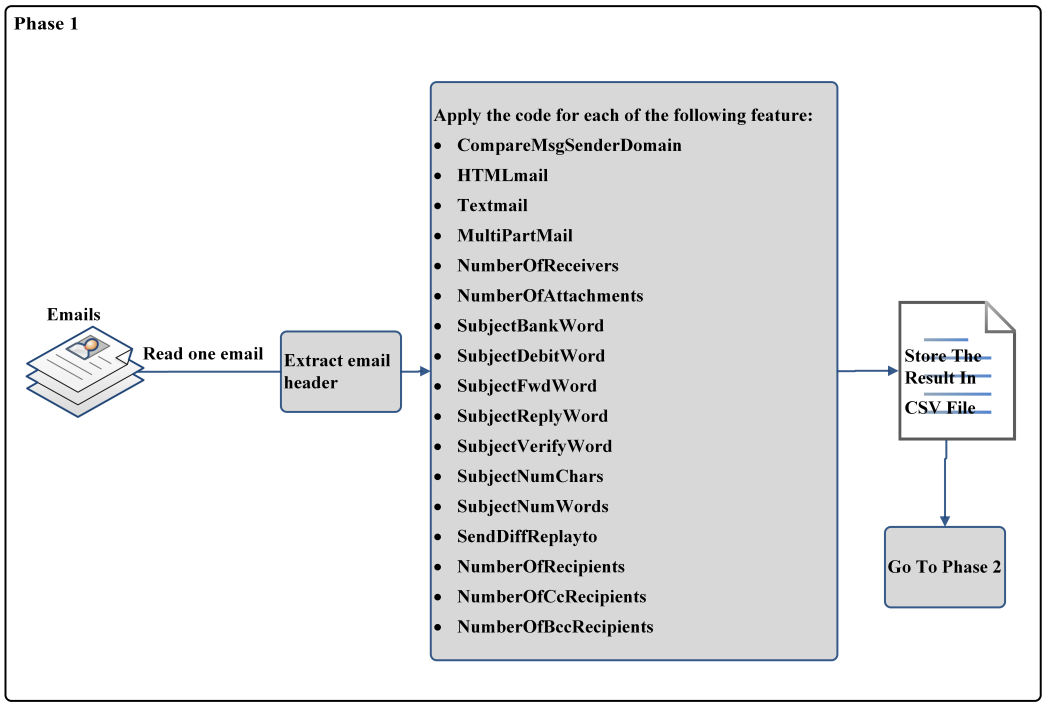


Figure 3.4: Pre-processing phase 1

In the first phase, as shown in Figure 3.4, the email header is extracted and apply the code for features (2, 3, 4, 22, 23, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 47, 48, and 49). The results for every feature is stored in a Comma Separated Values (CSV) file which will be used as input to the classification algorithm after completing the feature extraction process. Then the program start phase 2 by checking if the email content type is text/plain to determine the features that will be extracted depending on the email content type. In phase 2, as shown in Figure 3.5, a set of features is extracted from the email content. The extracted set of features depends on the type of email content; if the content-type is not text/plain then more features related to HTML content are extracted and subsequently the features related to the text of the email are extracted. At the end of phase 2, hyperlinks that may exist in the email body are extracted, if the number of hyperlinks is zero then the rest of features are initialized to their default values and phase 3 is ignored. Otherwise the program proceeds to phase 3 to extract features related to hyperlinks which will speed up the extraction process.

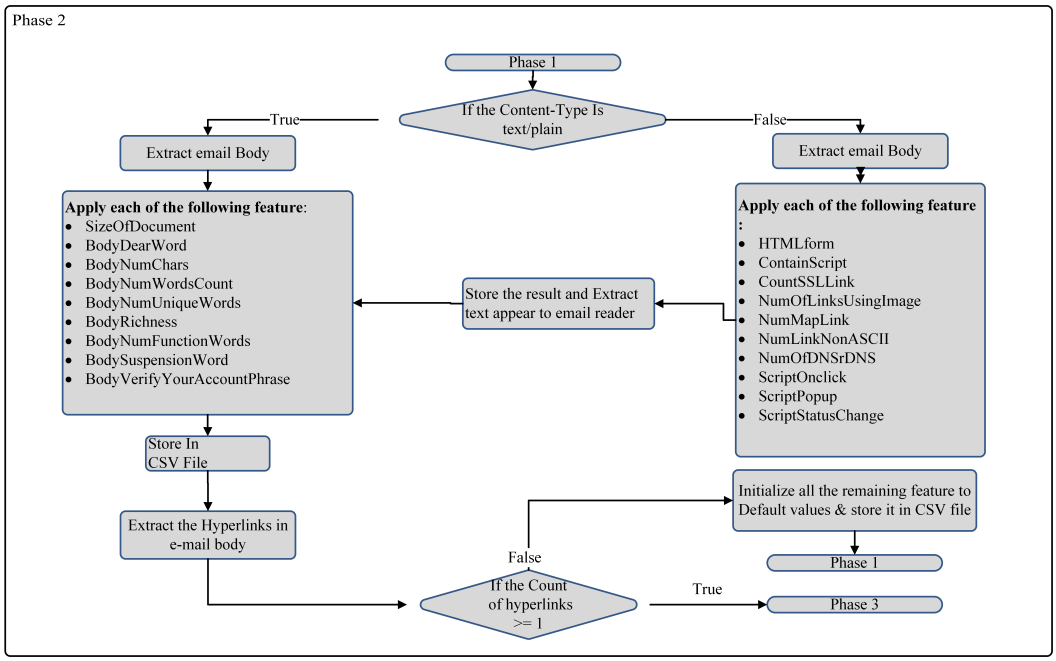


Figure 3.5: Pre-processing phase 2

All of these extracted features are stored in the same CSV file as in phase one, and then all of the hyperlinks located in the email body are extracted before proceeding to phase 3. As an example of how each feature is processed, Algorithm 1 shows the logic used for CompareMsgSenderDomain algorithm. The source code for this algorithm can be found in Appendix B.

Algorithm 1 CompareMsgSenderDomain

Input: Email
Output: Feature value

```
1: Read email header.
2: Extract the message id from email header.
3: Extract the sender from the email header.
4: Extract the domain of the email from message id
5: Extract the sender domain from the sender email address
6: if messageDomain == senderDomain then
7: Return true
8: else
9: Return false
10: end if
```

In phase 2, extracting all hyperlinks available in the email body. If the email content does not contain any hyperlinks, initialized all the remaining features and proceed with next email. Otherwise the extraction process continues to phase 3 as shown in Figure 3.6. The rest of the features are related to the hyperlinks located in the email body, which is the most important information that phishers use to lead victims to their spoof websites.

Figure 2 shows the pre-processing algorithm that explain how the fifty features are extracted from the email main content, the steps of the pre-processing algorithm are explained previously as shown in Figure 3.4, Figure 3.5, and Figure 3.6. the pre-processing algorithm takes as input the list of emails collected in the dataset and produces a CSV file which will be taken in later steps as input to train and test the

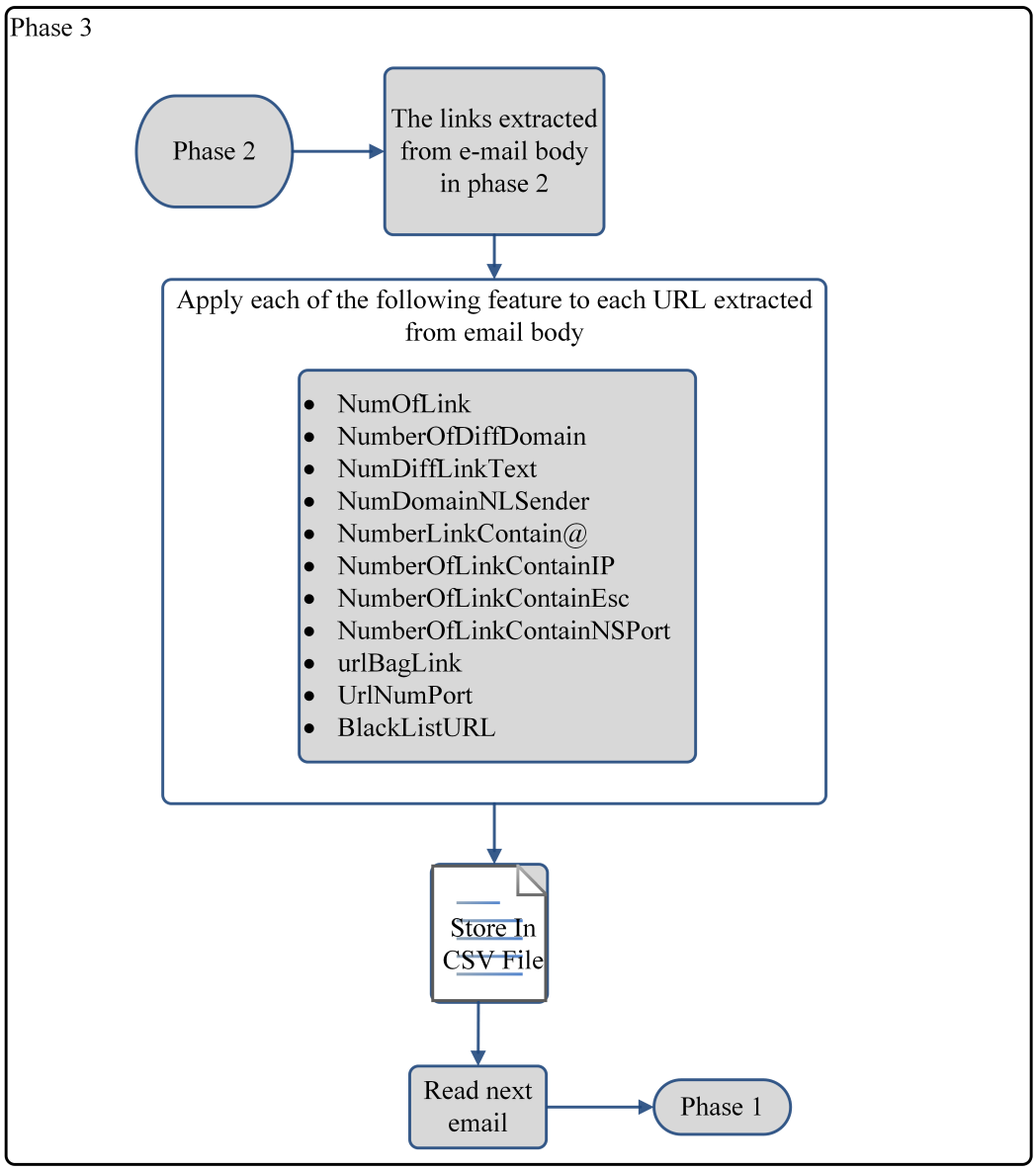


Figure 3.6: Pre-processing phase 3

proposed model. In the proposed algorithm the feature name is replaced by feature ID, these IDs are shown previously in Table 3.1, 3.2, 3.3, and 3.4.

3.5 Feature Evaluation and Reduction (FEaR)

After applying the pre-processing phase and extracting the fifty features described in the previous section, the results are sent to the FEaR algorithm. This algorithm is used to determine the number of actual features that the classification process is

Algorithm 2 The pre-processing algorithm

Input: Email

Output: 50 features extracted from each email stored in a CSV file

1: Read one email at a time.
2: Extract the email header.
3: Apply the code of each of the following features to the email header (1, 2, 3, 4, 22, 23, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 47, 48, and 49).
4: Save the result in the CSV ïňĄle, where every email has one record and every feature has one column.
5: **if** ContentType == "text/plain" **then**
6: Extract the email body, and apply each of the following features (20, 24, 25, 26, 27, 28, 29, 30 and 31). Initialize the default values for features (5, 15, 16, 17, 18, 19, 21, 44, 45, and 46) and go to step 7.
7: **end if**
8: Apply the following features to the email body (5, 15, 16, 17, 18, 19, 21, 44, 45, and 46). Then extract the text which appears to the email reader and apply features (20, 24, 25, 26, 27, 28, 29, 30 and 31) and store the results in the output file.
9: Collect all hyperlinks available in the email body and store them in a LINKS array.
10: **if** the size of the LINKS array == 0 **then**
11: Store the default values for all of the remaining features and update the CSV ïňĄle, and go to step 10.
12: **end if**
13: Apply features (6, 7, 8, 9, 10, 11, 12, 13, 14, 42, 43, and 50) to the list of hyperlinks (LINKS), then store the result in the output file.
14: **if** there is any more email **then**
15: go to step 1
16: **else**
17: go to step 11.
18: **end if**
19: Generate the dataset in CSV file format.

applied to. The selected features will depend on the training dataset, which is used to determine the actual number of features that is necessary to classify the emails in the training dataset. The features used will change if the training dataset is changed. Depending on the dataset used to build the detection model, the algorithm will choose from the fifty features a set of features that is effective in determining email type. The rest of the features are not considered important and will be excluded from subsequent analysis, which will speed up the development process while preserving the same level of accuracy as if the detection model was developed with all fifty features. The steps

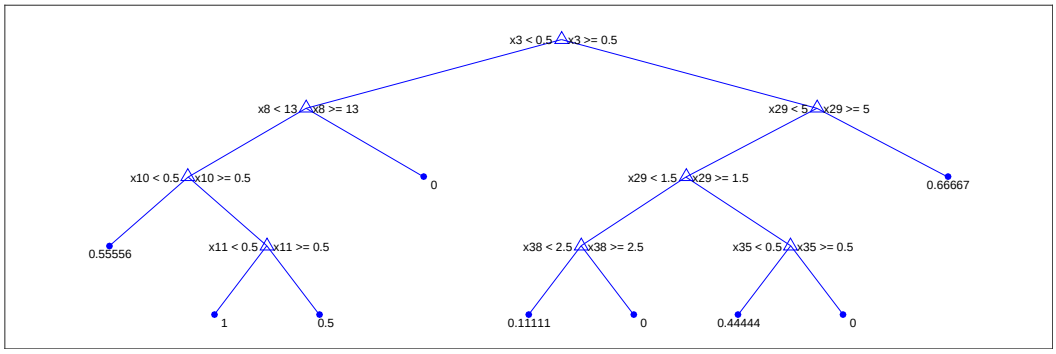


Figure 3.7: Regression tree for 200 emails generated by CART algorithm

of the FEaR algorithm is shown in Algorithm 3, this algorithm will evaluated the training dataset to determine the important list of features and exclude the irrelevant ones. Figure 3.7 shows an example of the regression tree generated by the CART algorithm (Berk, 2016), the steps of the CART algorithm is described in the first step of the FEaR algorithm as shown in Algorithm 3. The Cart algorithm start by choosing from the list of features the one that have best division in terms of email class for the training dataset. The data is divided based on the selected feature in the first step, the same process is repeated until the training dataset is best classified. For example for 200 email feature x_3 (Textmail) is chosen as the root of this tree, and the attribute value $x_3 = 0.5$. The same process is repeated until classify all email in the training dataset, in the generated regression tree the email is classified as a phishing if $x_3 \geqslant 0.5$ and $x_{29} \geqslant 5$. The leave of the regression tree will be the email class if the final decision is greater than 0.5 then the email class is phishing, otherwise the email class will be a legitimate email.

Table 3.5 shows an example of applying the FEaR algorithm to determine the important list of features that can be used to classify 200 emails. These values are estimated using FEaR algorithm for the regression tree shown in Figure 3.6. The feature impor-

Algorithm 3 FEaR

Input: Training dataset

Output: List of important features

1: Building a classification and regression tree using the CART algorithm. A brief description of how the CART algorithm works is as follows:

   a. Start with the training dataset.

   b. Take into consideration all possible values of all features; in this case, fifty features.

   c. Choose one feature x_i with a specific value t_1, $(x_i = t_1)$ that generates the largest division in the email class, which will be the feature that can have the best division in relation to the email class.

   d. Split the data into two branches: if $(x_i < t_1)$, then send the data to the left-hand branch; otherwise, send the data to the right-hand branch.

   e. Repeat the same process on these two âĂIJnodesâĂİ. The final output is a tree as shown in which is used as input to the second step. The nodes in this tree will be the selected features that can generate the best division of the training dataset.

2: Evaluate the importance of every feature, where every node in the tree generated in the first step will represent one feature. The importance of every feature (node 1) that has two children (nodes 2 and 3) is estimated by applying Equation 3.1:

$$V_1 = \frac{(R_1 - R_2 - R_3)}{Num\_node} \tag{3.1}$$

   where R_1, R_2 and R_3 are the node risks for the parent and children nodes, and Num_node is the total number of nodes in this tree. The risk is defined as shown in Equation 3.2.

$$R_i = P_i * E_i \tag{3.2}$$

   where P_i is the node probability and E_i is the node error computed by the CART algorithm for every node in step 1.

3: Create a crisp value for each feature by dividing the importance value of each feature computed in step 2 by the maximum importance value as shown in Equation 3.3:

$$\left\{ V_i = \frac{V_i}{max(v))} * 100 | \forall i = 1, 2, 3...n \right\} \tag{3.3}$$

   where V_i is the importance of feature i computed in step 2, n is the number of features, v is the vector of all features, and $max(v)$ is the maximum value in the vector v.

4: Select the features with crisp values $V_i > 0$ as important features. The features with $V_i = 0$ do not have a corresponding node in the tree constructed in step 1.

5: Reduce the number of features extracted in the pre-processing phase from 50 features to the list of important features determined in step 4.

tance is calculated using Equations 3.1, 3.2, and 3.3 will always be positive, because the CART algorithm will divide the dataset into two parts at every node. When dividing the dataset based on the selected feature the sub dataset at the children node will be part of the dataset that classified at the parent node. The sum of the dataset for the two children node will be less than or equal of the parent dataset. The important of every node (V_i) will always be greater than or equal to the sum of the two child nodes.

Table 3.5: An Example for applying the FEaR algorithm for 200 emails

Feature	P_i	E_i	R_i	$(R_1 - R_2 - R_3)$	V_i	$Rank_i$
3	1	0.2491	0.2491	0.1552	0.0194	100.0000
8	0.525	0.0862	0.0452	0.0215	0.0027	13.8427
29	0.475	0.1024	0.0486	0.0173	0.0022	11.1744
10	0.5	0.0475	0.0238	0.0077	0.0010	4.9564
29	0.43	0.0548	0.0235	0.0173	0.0022	11.1744
11	0.455	0.0109	0.0049	0.0024	0.0003	1.5751
38	0.32	0.0154	0.0049	0.0005	0.0001	0.3076
35	0.11	0.1488	0.0164	0.0053	0.0007	3.3837

3.6 Offline Phishing Email Detection System

Figure 3.8 shows the general components of the proposed model, which is designed to evaluate the ability of the proposed pre-processing algorithm and FEaR algorithm to classify email with high accuracy. Ten different classification algorithms have been used to train the proposed model. These algorithms are the BayesNet (Hamid and Abawajy, 2011), SimpleCART (Abu-Nimeh et al., 2007), J48 (Gansterer and Pölz, 2009), Decision Table (Chandrasekaran et al., 2006), MultilayerPerceptron (Ma et al., 2009), NaiveBayes (Ma et al., 2009), PART, Random Forest (Fette et al., 2007), SMO and Logistic Regression (Ma et al., 2009). A comparison has been conducted to determine the best algorithm that can be used for this purpose. The classification algorithms

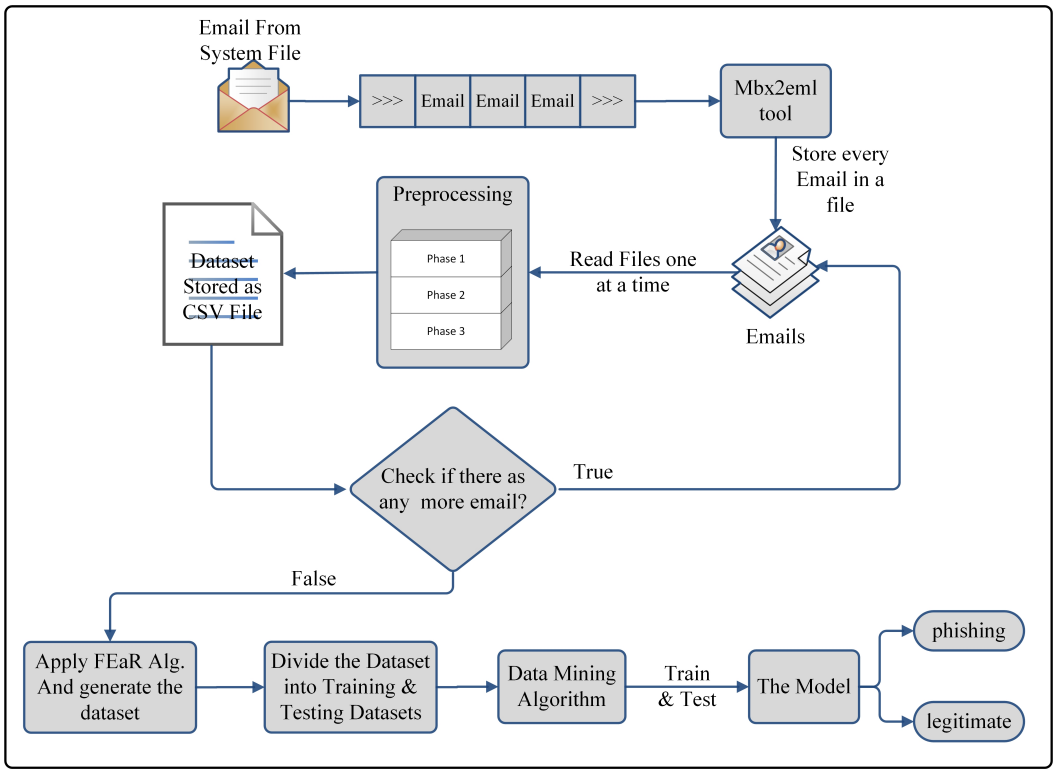


Figure 3.8: Offline phishing email detection system

selected from the literature are the most commonly used algorithms in such a classification model. The model consists of mbx2eml, which is a free tool used to split emails grouped in the mbox file format and each email is stored in a separate file. Then the pre-processing of the dataset is conducted to extract 50 features.

Further training and testing experiments were performed using a 10-fold cross-validation method based on data mining algorithms, in which system accuracy and robustness are measured. Cross-validation is used because the cross-validation estimator gives better results than a single hold-out set estimator, which is very helpful if the size of the dataset available is limited, as in this thesis. If a single fold out test is used, where 90% of the data is used for training and 10% for testing, then the test set is very small, and so it will not represent all kinds of phishing email. The performance estimate will change if the experiment is repeated with different groups of data to form the training

and test sets. The 10-fold validation reduces these changes in performance by taking an average of 10 different partitions. Therefore, the performance estimate is less sensitive to the partitioning of the dataset (Tsamardinos et al., 2014). The experimental results show that the proposed model with hybrid features gives the best performance compared with all previously reported approaches.

3.7 Experimental Results and Discussion

3.7.1 Dataset

Three publically available datasets were used; two of them are email datasets and one is of phishing URL datasets. These datasets are described as follows:

- PhishingCorpus (Nazario, 2015) The PhishingCorpus is a phishing email dataset. This dataset was collected manually, and researchers in this field have used this dataset extensively. The PhishingCorpus was collected from 2004-2007, and the most recent update was in 2015, where the total number of emails collected was 7315.

- SpamAssassin (Mason, 2005) The SpamAssassin project collected a total of 6047 of which 4951 are ham emails. In the training and testing of the proposed model, the ham email used is from the SpamAssasin corpus.

- PhishTank (OpenDNS, 2016) The PhishTank URLs dataset was collected by the PhishTank organization, which is a collaborative clearing house for data and information about phishing on the Internet. This dataset is used to update the content of the BlackListURL feature, and the list of blacklisted URLs are

automatically updated every 60 minutes by the PhishTank website. Total of 26722 phishing URL were collected, and this number is updated every 60 minutes.

3.7.2 Technical terms in detection

Throughout this thesis, when evaluating phishing detection, many different technical terms from signal detection theory are used. As a point of reference, this section gives a short introduction exploring these properties.

Suppose that M denotes the total number of phishing emails and D denotes the total number of legitimate emails. Then $nm \in M$ is the number of correctly detected phishing emails, while $nd \in D$ is the number of emails correctly detected as legitimate, nf is the number of legitimate emails detected as phishing, and np is the number of phishing emails detected as legitimate. For each detected email, the following evaluation method is applied: True positive (TP): the number of phishing emails correctly classified as phishing.

$$TP = nm/M \tag{3.4}$$

True negative (TN): the number of legitimate emails correctly classified as legitimate.

$$TN = nd/D \tag{3.5}$$

False positive (FP): the number of legitimate emails incorrectly classified as phishing.

$$FP = nf/D \tag{3.6}$$

False negative (FN): the number of phishing emails incorrectly classified as legitimate.

$$FN = np/M \tag{3.7}$$

Based on these four rules, other factors can be derived: precision, sensitivity, accuracy, and F_Measure (Ferri et al., 2009; Wickens, 2002; Zhu et al., 2010). Accuracy is the sum of TP and TN (the number of correct decisions) divided by the total number of emails. Precision is the ratio of how many decisions were correct, which is TP divided by the sum of TP and FP. Sensitivity is the number of TP assessments divided by the number of all positive assessments. The $F\_Measure$ is a measure of the test's accuracy, which refers to the balance between precision and sensitivity. These four metrics are shown in following definitions:

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{3.8}$$

$$Sensitivity = \frac{|TP|}{|TP| + |FN|} \tag{3.9}$$

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{3.10}$$

$$F\_Measure = \frac{2 * Precision * Sensitivity}{Precision + Sensitivity} \tag{3.11}$$

In the assessment of any phishing email detection system, the goal will be to increase the percentage of true positive and to keep the percentage of false-positives close to zero. The false negative and true negative rates are complementary values that can be calculated depending on the values of TP and FP respectively.

3.7.3 Feature selection using the FEaR algorithm

To evaluate the ability of the FEaR algorithm to detect new behaviour available in the dataset, many experiments were conducted. After applying the pre-processing algorithm and extracting the fifty features for all email in the selected dataset (9900 emails), the dataset is divided into a sub dataset, the element of each dataset were

chosen randomly and used as input to the FEaR algorithm to explore the important list of features.

As shown in Table 3.6, for a dataset with 500 emails, the FEaR algorithm detected 11 features as important features that can be used to distinguish between phishing and ham emails. With changing in the training dataset, a new phishing email is explored which could mean a new behaviour used by the phisher to lure online customers. The FEaR algorithm builds a different regression tree that can best classify the emails in the training dataset, and in every experiment it changes the tree nodes (features) from the total list of features. The FEaR algorithm showed the ability to explore these behaviours, when the algorithm processes datasets of different list of emails, a different list of features is selected. In the next section, the selected list of features is used to build the classification model, and the system is evaluated to see if it can detect phishing emails with high accuracy. In Table 3.6, for the sub dataset of size 1000, the FEaR algorithm explore different set of features, this happen due to the fact that different set of email where chosen randomly where different phishing email may include different behaviours user by phishers to lure the online customers. The ability of the proposed algorithm to dynamically choose different set of behaviours when the training dataset is changed will be main part of the online phishing email detection system proposed in Chapter 5.

The proposed algorithm for the selection from a large set of features solves the problem of selecting the right set of features to be used to build a classification model. Furthermore, the number of important features changes dynamically when the dataset changes without any user intervention, whereas other studies (Del Castillo et al., 2007;

Table 3.6: Important features discovered using FEaR algorithm

Dataset Size	**Count of Important Feature**	**Features**
500	11	3, 6, 11, 20, 24, 28, 29, 35, 37, 39, 41
1000	17	1, 3, 6, 7, 11, 17, 20, 25, 26, 28, 29, 34, 35, 37, 39, 40, 44
1000	18	3, 6, 7, 11, 20, 21, 24, 25, 26, 27, 28, 29, 35, 37, 38, 39, 40, 41
1000	18	3, 6, 7, 8, 10, 11, 20, 24, 25, 26, 28, 29, 34, 35, 37, 39, 40, 41
1500	19	3, 6, 9, 10, 11, 17, 20, 22, 23, 24, 26, 27, 28, 29, 35, 37, 38, 39, 40
2000	23	1, 3, 6, 7, 10, 11, 16, 17, 20, 22, 24, 26, 27, 28, 29, 34, 35, 36, 37, 39, 40, 41, 42
2500	27	1, 3, 6, 7, 8, 11, 16, 17, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 34, 35, 36, 37, 38, 39, 40, 41, 48
3000	28	3, 4, 6, 7, 9, 10, 11, 12, 20, 22, 23, 24, 25, 26, 27, 28, 29, 32, 35, 36, 37, 38, 39, 40, 41, 42, 43, 48
3500	29	1, 3, 6, 7, 8, 10, 11, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44
4000	30	1, 3, 6, 7, 8, 9, 10, 11, 16, 17, 20, 22, 23, 24, 25, 26, 27, 28, 29, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44
9900	33	1, 3, 6, 7, 8, 9, 10, 11, 16, 17, 18, 20, 22, 23, 24, 25, 26, 27, 28, 29, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 48

Gansterer and Pölz, 2009; Hamid and Abawajy, 2011; Islam and Abawajy, 2013; Pamunuwa et al., 2007; Salem et al., 2010) have used fixed numbers of features to classify phishing email and the sets of features chosen are changed manually. Moreover, the proposed algorithm can be used to rank the selected features, as shown in Table 3.7 where the ranks V_i are evaluated in step 3 in the FEaR algorithm. In the online mode (to be discussed later in Chapter 5), when building a new dataset that contains new phishing emails, the FEaR algorithm will explore the new behaviours that the new dataset may contain.

Some of the newly proposed features are selected by the FEaR algorithm. As shown in Table 3.6, features (3, 4, 22, 23, and 48) are the new features proposed in this study are selected as important features by FEaR algorithm. The selection of these new features by FEaR algorithm depends on the size of the dataset, as an example, if the dataset size is 500 email, then the FEaR algorithm select one new feature (feature 3). For 3000 email Features (3, 4, 22, and 23) are selected as important. The rank of the

Table 3.7: Feature ranking evaluated by FEaR algorithm with 4000 emails

Feature ID	**Rank**	**Feature ID**	**Rank**
3	100	25	0.663486
7	13.96487	17	0.546267
24	6.499948	11	0.498457
35	5.17843	22	0.387983
26	3.771794	23	0.346812
29	2.266412	1	0.341654
28	2.033311	16	0.295951
6	1.810769	38	0.28526
39	1.278216	32	0.282259
41	0.926993	34	0.214601
20	0.914429	36	0.13632
27	0.878601	10	0.06674
44	0.829632	42	0.045931
37	0.702157	8	0.036652
40	0.684341	9	0.036301

newly proposed features shows that such features are part of most important features when compared with the previously proposed features as shown in Table 3.7. As an example, feature 3 is selected as the most important feature.

In this section, the FEaR algorithm is evaluated with different sizes of dataset using the complete list of features as an input and selecting the most effective list of features that can classify emails in the dataset under investigation. If the dataset size changes, it contains different groups of phishing and ham email. The FEaR algorithm successfully selects different features lists for different datasets. The experiments conducted in the next section shows that the selected list of features by FEaR algorithm can be used to build a classification model that classify emails with a high level of accuracy.

3.7.4 Experimental setup

The following the experiment is repeated with different classification algorithms used to train the classification model. In all experiments approved datasets were used (see Section 3.7.1 for a full description of the dataset used). The phishing email dataset

suggested by Nazario (Nazario, 2015) has been used in many research studies such as (Hamid and Abawajy, 2011; Ma et al., 2009; Toolan and Carthy, 2009), and for legitimate emails the SpamAssassin project (Mason, 2005) datasets were used.

The results of the experiment show that the pre-processing phase has the most influence on the results for all metrics, when compared with previous studies that used the same classification algorithm to build the detection model for the same datasets. Table 3.8 shows the results for the proposed model with a set of algorithms that have been used in the literature to solve such problems. Our experiments lead to high accuracy and true positive rates, with low false positive rates compared to the results of other studies.

3.7.5 Comparison of the performance of different classification algorithms

In the literature, various classification algorithms have been used to classify emails into phishing and ham email. In this study, 10 different classification algorithms drawn from previous studies were compared, and these represent the most widely used algorithms in classification for similar problems. A comparison was performed on the same dataset of 9900 emails. The results of this experiment are summarized in Table 3.8.

3.7.6 Results and discussion

The result for the classification algorithm that gives the best outcome according to the chosen metrics, as shown in Figure 3.9, is the Random Forest algorithm. As shown in Figure 3.9, the highest TPR is 97.98% for Random Forest, followed by 97.31% for the

Table 3.8: Classification results for 10 classification algorithms

Algorithm	**TPR**	**TNR**	**FPR**	**FNR**	**ACC**	**Precision**	**Sensitivity**	F-Measure	**ROC**
Random Forest	97.25%	98.71%	1.29%	2.75%	97.98%	98.69%	97.25%	97.96%	98.90%
SMO	92.51%	97.84%	2.16%	7.49%	95.17%	97.72%	92.51%	95.04%	95.40%
BayesNet	88.73%	98.00%	2.00%	11.30%	93.36%	97.80%	88.73%	93.04%	98.40%
SimpleCART	96.85%	97.78%	2.22%	3.15%	97.31%	97.76%	96.85%	97.30%	98.30%
J48	96.57%	97.82%	2.18%	3.43%	97.19%	97.79%	96.57%	97.17%	98.20%
Logistic Regression	93.49%	98.10%	1.90%	6.51%	95.80%	98.01%	93.49%	95.70%	98.10%
Decision Table	92.22%	97.98%	2.02%	7.78%	95.10%	97.86%	92.22%	92.22%	95.70%
Multilayer-Perceptron	94.85%	97.27%	2.73%	5.15%	96.06%	97.20%	94.85%	96.01%	98.30%
NaiveBayes	89.39%	90.55%	9.45%	10.60%	89.97%	90.44%	89.39%	89.91%	90.20%
PART	96.83%	97.70%	2.30%	3.17%	97.26%	97.68%	96.83%	97.25%	98.50%

SimpleCART algorithm. The highest accuracy level found was 97.89% for the Random Forest algorithm as shown in Figure 3.9, and the second highest accuracy was 97.31% for the SimpleCART algorithm. The lowest registered result for NaiveBayes algorithm for all metrics since it need a large dataset in order to make reliable approximations of the probability of each class, which is not available for phishing email detection. The

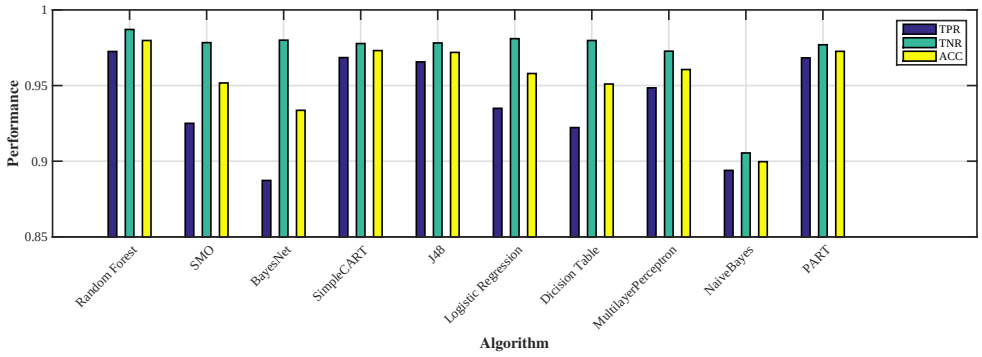


Figure 3.9: Comparison of classification algorithms in term of TPR, TNR, and accuracy

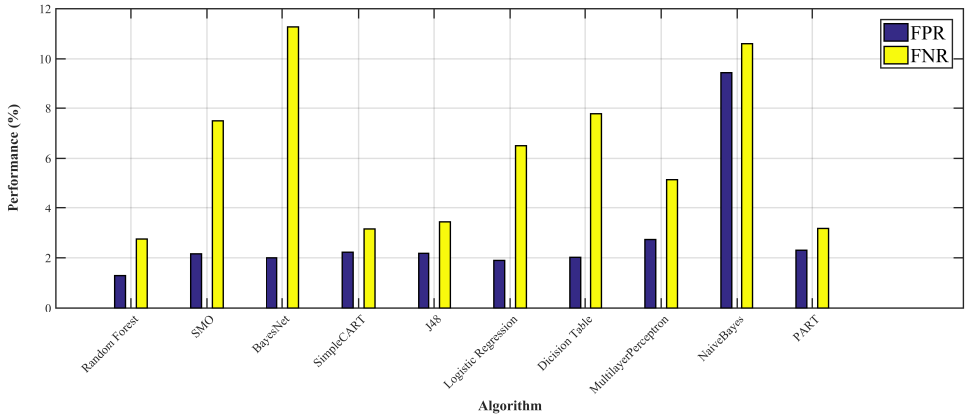


Figure 3.10: Comparison of classification algorithms in term of FPR and FNR

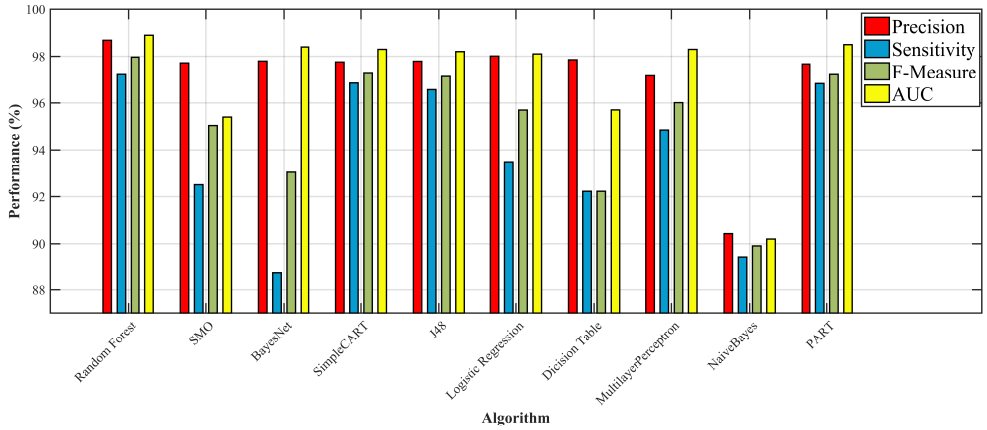


Figure 3.11: Comparison of classification algorithms in term of Precision, Sensitivity, F-Measure, and area under ROC curve

lowest FPR was registered for the Random Forest algorithm at 1.29%, followed by the Logistic Regression algorithm as shown in Figure 3.10. Furthermore, the highest area under the receiver operating characteristic (ROC) curve (AUC) was registered for the Random Forest algorithm as shown in Figure 3.11. The ROC measure helps to indicate graphically the performance of a binary classifier system as its discrimination threshold is varied. The classification algorithm that shows the best result according to the chosen metrics as shown in Table 3.8 are the Random Forest and SimpleCART algorithms as shown in Figure 3.9, 3.10, and 3.11. Random Forest has the best result

because of its nature, Random Forests built different decision trees to classify a new mail from an input dataset, Random Forests put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree votes for that class. The forest chooses the classification tree having the most votes (Breiman, 2001). Random Forest algorithm has a lot of advantages that make it one of the most powerful classification algorithm, for example it runs efficiently on small datasets where other algorithm such as NaiveBayes needs a very large training dataset to generate a classification model with an acceptable error rate. Moreover, it can handle a massive number of features without ignoring any of them. It can estimate missing data which is a common problem in this field, where it maintains accuracy when a large proportion of the data are missing. The missing data problem is addressed well through the implementation of pre-processing algorithm by insuring that every feature is initialized with a default value if it is not available in the email under processing. The advantages of the Random Forests algorithm were discussed in more details by Boulesteix et al. (Boulesteix et al., 2012).

3.7.7 Comparative analysis

The performance of the proposed offline phishing detection model is compared with previous studies that make an attempt to solve the phishing detection problem. To perform the comparison and discuss the advantages and disadvantages of every study, many criteria should be taken into consideration such as, is the dataset used to assess the proposed technique benchmarked? Did other researchers use the dataset? What is the dataset size? Is it balanced or imbalanced dataset? What are the metrics used in the system evaluation (one metric will not be adequate to evaluate the system

performance)? What are the evaluation methods? Based on the previous questions the best model will be the one that used benchmark dataset with an acceptable size for testing. The metrics used, depending on the dataset type (balanced or imbalanced). Moreover, the validation method that makes the result authentic.

Table 3.9 shows the outcome of the comparison of our results with those of previous work. Khonji et al. (2012) used 47 features to represent phishing emails and achieved 97% accuracy (Khonji et al., 2012). However this study did not take into consideration metrics such as true positive, true negative rate, AUC, and false negative rate, despite the fact that these metrics are extremely significant. Gansterer and Pölz (2009) used 30 features, and applied the J48 and SVM algorithms for classification and achieved 97% accuracy (Gansterer and Pölz, 2009). However, this study depended on a large number of online features (15 feature) and the extraction of too many online features may affect the performance and scalability of the email filtering system, the use of online features increases the pre-processing time and it highly depends on other criteria like the Internet bandwidth. Chandrasekaran et al. (2006) used a very small dataset to test the system with 200 emails for phishing and 200 emails for legitimate emails, which could not fully represent all phishing emails (Chandrasekaran et al., 2006).

Meanwhile Ma et al. (2009) proposed a hybrid approach using seven features. The authors claim achieving 99% accuracy (Ma et al., 2009). What casts doubt though on this accuracy is that a highly imbalanced dataset was used where only 7% of emails are phishing. Additionally, accuracy is the only metric used to assess the proposed model where such cases need other metrics to assess the proposed model such as Matthews Correlation Coefficient (MCC) (Matthews, 1975) and the area under the ROC. This is

because if the dataset used is imbalanced (99 to 1), and the system classifies all emails as legitimate, then the accuracy will be 99%. Moreover, the used dataset in this study was collected from live emails received by WestPac only; which has not been used by other studies and not benchmarked.

Abu-Nimeh et al. (2007) examined 43 keywords to detect phishing emails, and the study achieved 94.5% accuracy (Abu-Nimeh et al., 2007). On the other hand, this study ignored features related to hyperlinks, which are the most important information related to phishing attacks.

Toolan and Carthy (2010) used 22 features to test three datasets (Toolan and Carthy, 2010). Approximately 97% accuracy was achieved for the first test which did not include phishing emails, but for tests 2 and 3 which included phishing emails much lower accuracy rates of 84% and 79% were achieved. Hamid and Abawajy (2011) also used seven hybrid features with several datasets and achieved 96% accuracy, but this accuracy rate was reduced when dataset size increased (Hamid and Abawajy, 2011). Our approach used 33 hybrid features selected using the FEaR algorithm, and 97.98% accuracy was successfully achieved with a FP rate of 1.29% for the Random Forest algorithm. The results for other metrics are summarized in Table 3.1. The proposed pre-processing algorithm extracted 50 features that contain information from all parts of emails that may used by phishers to trick online customers. Based on the dataset used, the FEaR algorithm used the 50 features as input and built a regression tree that best classified the training dataset. The list of features selected by the FEaR algorithm was used by the classification algorithm to build the classifiers that were later employed to classify emails into phishing and ham email. The pre-processing

Table 3.9: Comparison of our approach with previous works

Author	Features	Algorithm used	Feature approach	Sample	Results
Khonji et al. (2012)	47	Random Forest	Hybrid	Phishing 4116 Ham 4150	Accuracy 97% FP 0.60%
Gansterer and Pölz (2009)	30	J48 SVM	Hybrid	Phishing 5000 Ham 5000	Accuracy 97%
Chandrasekaran et al. (2006)	25	SVM	Content	Phishing 200 Ham 200	Precision 100% Recall 50% Accuracy 75%
Ma et al. (2009)	7	DT, Random Forest, Multi-layer perception, NaiveBayes, SVM	Hybrid	Phishing 46,525 Ham 613,048	Decision Tree Accuracy 99%
Abu-Nimeh et al. (2007)	43	LR, CART, SVM, NN, BART	Keyword-based	2889	Accuracy 94.5%
Toolan and Carthy (2010)	22	C5.0	Hybrid	Ham 4202 Spam 1895 Phishing 4563	Accuracy Test 1 (97%) Test 2 (84%) Test 3 (79%)
Hamid and Abawajy (2011)	7	Bayes Net	Hybrid	Corpus (1) 1645 Corpus (2) 2495 Corpus(3) 4594	Accuracy for Corpus (1) 96% Corpus (2) 92% Corpus (3) 92%
Proposed approach	33	Random Forest	Hybrid	Phishing 4950 Ham 4950	Accuracy 97.98% FP 1.29%

and FEaR algorithms in the proposed model increased the overall system performance when compared with other strategies that used the same classification algorithm and the same dataset.

3.7.8 Other finding

Experiments were conducted with ten different classification algorithm to show which of them give the best result in terms of the chosen metrics. The best algorithm that produces the best result for all metrics is Random Forest as shown previously in Table 3.8. Some of previous studies such as Gansterer and Pölz (2009) highly depend on online feature which need an extra information loaded from the Internet every time a new email is pre-processed (Gansterer and Pölz, 2009). In the proposed model we eliminate the need of online features where we have only one online feature (BlackListURL), the database of blacklisted URLs used for this feature is updated every 60 minutes in an independent way of the pre-processing of email which will not slow down the pre-processing system.

3.8 Summary and Conclusion

This chapter has presented an offline model for email classification into legitimate and phishing emails based on hybrid features which depend on the information extracted from the email headers and content. Data mining algorithms were used to build the detection mechanism. Using 10-fold cross-validation, the overall results show that the proposed model with 33 features gives better results than previously published results, where the highest accuracy registered is 97.98% for the Random Forest algorithm.

One might wonder that features would be different for each algorithm. It should be

emphasised that the selection of the best classification algorithm is independent from the feature selection. Since selection of the important feature is done using FEaR algorithm before the model being trained and tested using one of the classification algorithm as shown in the system model in Figure 3.8.

The pre-processing algorithm extracted a large set of features which cover all aspects of emails where features are selected and extracted from the email header and content. Furthermore, the FEaR algorithm solve the problem of selecting the active list of features that can classify the training dataset with the highest accuracy.

The proposed model was tested with ten data mining algorithms selected from the literature, and an experiment was conducted to show the merits of the proposed pre-processing and FEaR algorithms as well as to determine the best algorithm that can be used for the detection of phishing emails. This hybrid feature selection approach produced promising results using 33 features, which were reduced from fifty features using the FEaR algorithm. The algorithm that shows the best result is Random Forest, when used to build the classification model with an accuracy rate of 97.98%, and the lowest false positive rate was registered for the Random Forest algorithm at 1.29%. This improvement is due to the pre-processing and FEaR algorithms proposed in this thesis.

In summary, the most significant and novel results in this experiment show that the extraction of features in the pre-processing phase has a most important influence on the outcome of the classification model. In addition, the study provides the highest results so far published in terms of accuracy and false positive rate for an approved dataset. On the other hand, the study used a wider range of metrics than previous

research, including true positive, true negative, false negative rates, the F-measure, ROC, precision, and sensitivity.

4

DYNAMIC EVOLVING NEURAL NETWORK USING REINFORCEMENT LEARNING

4.1 Introduction

This chapter discusses in detail the DENNuRL algorithm that will be the core of the phishing email detection model. An experiment is conducted to prove the ability of the DENNuRL algorithm to generate the best NN architecture that can be used in classification problems. To evaluate the performance of DENNuRL, a well known benchmark classification problem has been used, which is the Diabetes dataset (Bache and Lichman, 2013). The chapter ends with a comparison of standard techniques used to adapt NN to be used in a classification problems and the DENNuRL algorithm.

4.2 Artificial Neural Network

An ANN is a network of neurons connected to each other in a layered approach, and the idea behind the ANN comes from a simulation of how the human brain works. The ANN was first invented by McCulloch and Pitts (1943), but it could not be put into practice given the computing power available until the work by Werbos in 1974. McCulloch and Pitts (1943) developed the back-propagation algorithm (Zhang and Zhang, 1999), and since that time ANN have been used in many different applications such as robust pattern detection (Basu et al., 2010), signal filtering (An et al., 2013), data segmentation (Sowmya and Rani, 2011), data compression and sensor data fusion (El Faouzi et al., 2011), data mining and associative searching (Chou et al., 2010), adaptive control (Sun et al., 2011), modelling complex phenomena (Costa et al., 2011), and designing an adaptive interface for human-machine systems (Hsieh et al., 2010).

An ANN consists of one or more layers, where each layer contains one or more neurons, and the number of layers depends on the complexity of the problem to be solved. The ANN is trained by using feed-forward or back-propagation techniques. The most famous training algorithm is called the back-propagation algorithm where the error value is propagated from the output neuron to the input neuron, and the connection weight is modified to reduce the error the next time the NN is trained with other examples (Negnevitsky, 2005).

The NN is one of the most powerful techniques that has been used to build classification models. The ability of a NN model depends on many factors such as the number of layers, the number of neurons in the hidden layers, the training algorithm used, the number of training epochs and the problem to be solved. Usually, researchers deter-

mine the architecture of a NN manually, which often means that the model generated will not be optimal in solving the problem faced. Researchers have proved that a three-layer NN can solve any linear or non-linear problem (Aggarwal, 2007; Srivastava et al., 2014; Zhang and Zhang, 1999). The numbers of input and output neurons depend on the problem to be solved, and so the control of the number of neuron in the hidden layer will determine the power of the model generated using an ANN.

A multilayer feedforward ANN contains at least three layers, including an input layer, one or more hidden layers, and an output layer. With a fully connected ANN, all neurons in the hidden layer are connected to all neuron in the previous and next layers (see Figure 4.1). The input layer consists of a set of M neurons each of them is connected to one feature $(Z_1 - Z_M)$, and the output of each neuron in the input layer is connected to all neurons in the hidden layer N neurons the number of which depends on the complexity of the problem being solved. The same process is repeated to connect all neurons in the hidden layer with the output layer K neurons, the number of neurons in which depends on the number of classes in the proposed classification. The connections between neurons v_{nm} and w_{kn} have weightings which are modified during learning and fixed during the testing of the trained model (Negnevitsky, 2005). Training of the ANN is conducted by reading the inputs from the list of features at the input layer, and the output of every neuron is calculated depending on the training algorithm used. The output of every neuron is then propagated through the connections to the next layer. The same procedure is repeated until the output layer is reached. The error is computed for the training data depending on the difference between the expected output and the real output for every record in the training

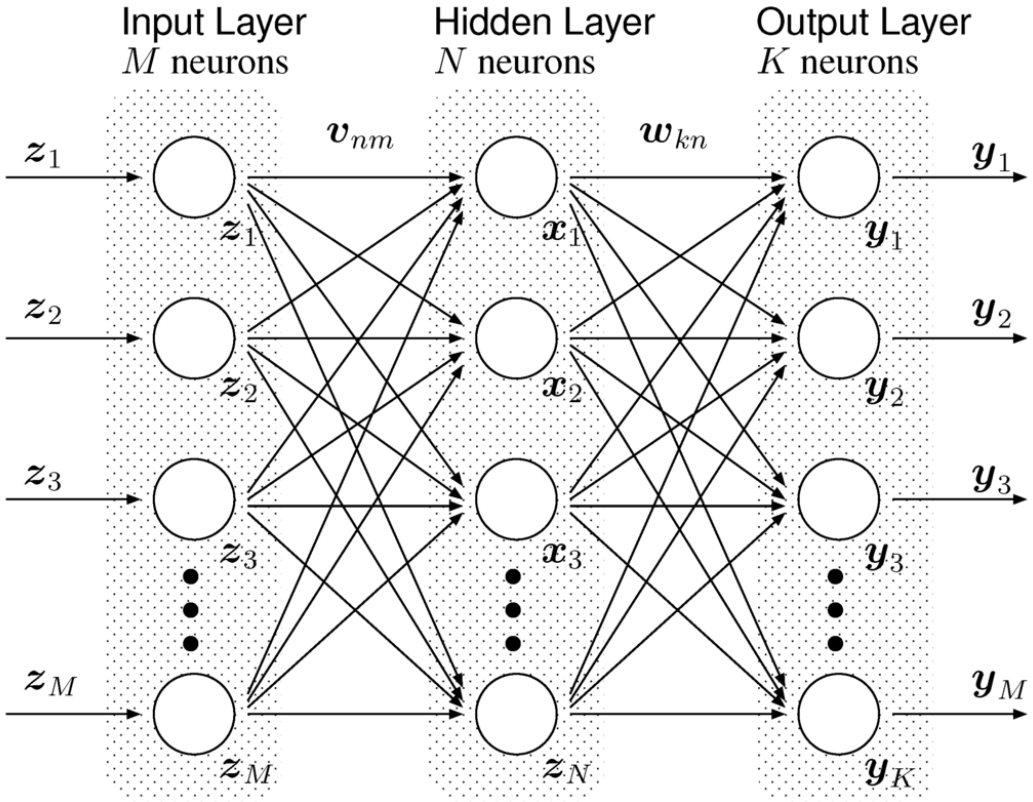


Figure 4.1: A feed-forward ANN with three layers: one input, one hidden, and one output layer (Negnevitsky, 2005)

dataset. Later, the connection weightings are modified to reduce the error computed in further training.

4.2.1 Neural Network Training

There are two types of ANN training methods: supervised and unsupervised training. With supervised training, the target for the training dataset is already known, but for unsupervised training the target is not known and the training depends on identifying the pattern in a group of features inorder to reach a stable state after a certain number of iterations (Svozil et al., 1997).

The most popular type of training algorithm for supervised training is called a back-propagation training algorithm, where the training process starts by initializing the

weighting and bias randomly with values between -0.5 to 0.5 (Negnevitsky, 2005). Adjustment of weight and bias could occur after every record in the training dataset, which is called incremental training, or modification could happen after the completion of the training dataset (which is also known as the training epoch) and this kind of training is called patch training (Gudise and Venayagamoorthy, 2003). The choice of which kind of training depends on the problem being solved. Each of these two methods includes a set of algorithms that have been proposed by researchers to speed up the training process and solve problems such as generalization, local minima, and overfitting. These kinds of challenges are common when NN is used to build classification models. ANN training can also be static or dynamic, as discussed in the next two sections.

4.2.2 Static neural network

The static NN can adapt weight and bias values only while training using the training dataset, and after the training process is completed the trained model generated remains unchanged. Well-known types of static NN are the multilayer perceptron, radial basis function networks, wavelet networks, and fuzzy networks.

4.2.3 Dynamic neural network

In a dynamic , or adaptive NN, the training continues during operation and is a continuous process. Well-known types of dynamic ANNs are recurrent neural networks (RNNs), and time-delay neural networks (TDNNs). In this thesis the ANN is considered to be the core of the classification model, and the choice of NN depends on the ability to change the generated model to reflect variations in the environment so as to

handle zero-day phishing attacks.

4.3 Common Techniques used to Build Neural Networks

In the design process and the choice of which NN architecture is better for a specific problem, many approaches have been used which include constructive Constructive approach (CA), Pruning approach (PA), Constructive-pruning approach (CPA), and Evolution approach (EA). These techniques target the number of neurons in the hidden layers, because the number of neurons in the input and output layers depend on the problem being solved. The next four sections briefly discuss these types of solutions.

4.3.1 constructive approach

An NN built using a constructive technique starts the development process by choosing a small number of neurons in the hidden layer. The NN generated is trained and tested, and if the termination condition is satisfied then the adaptation process ends and the last NN is chosen as the classification model. Otherwise the number of neurons in the hidden layer is incremented by one, and the new NN is then trained. The new NN is tested, and if it reaches the termination condition then the process stops; otherwise the incrementing process continues. Many researchers have used the constructive approach to choose the best NN architecture (Kwok and Yeung, 1997; Parekh et al., 2000; Wang et al., 2015). The main advantages of using this technique are firstly that it is easy to program. Secondly, the complexity of the model generated using this technique is always the best solution because this approach always searches for the NN with

the lowest number of neurons. Finally, it is straightforward to specify an initial NN architecture. The main disadvantage of using this technique is that it can become trapped in local minima where it cannot discover a better solution if adding one neuron cannot reduce the error (Kwok and Yeung, 1997).

4.3.2 pruning approach

Pruning techniques to build NNs start by selecting a large number of neurons in the hidden layer. The generated NN is trained and tested, and then the number of neurons and/or connections is reduced. The newly generated NN is then trained and tested, and if the new NN has a lower error rate then it replaces the old NN (Dai, 2013; Han and Qiao, 2010; Yu et al., 2011). The pruning process continues until no more enhancements can be achieved, and the final NN architecture is selected as the classification model for the problem at hand. To select which neurons and/or connections are to be removed, the algorithm estimates their importance and selects the least important for removal. The main advantages of using pruning techniques compared to a constructive approach is that they have the ability to arrive at the optimal case sooner. In addition, there is less chance of entrapment in local minima. However, the disadvantages from pruning algorithms include that it is hard to decide how big the initial NN should be. Furthermore, when compared with constructive approach, it involves more complex computation. Additionally, it is still not completely safe from being trapped in local minima.

4.3.3 Constructive-pruning approach

This technique combines the two previous techniques to avoid the disadvantages of each. CPA start by applying the CA, and if the generated NN architecture becomes too large, the PA refines the NN architecture. This technique has been used by many researchers (including Hirose et al., (1991), Islam et al., (2000), Han et al., (2007), Puma-Villanueva et al., (2012), Puma-Villanueva et al., (2012), and Yang & Chen, (2012) (Han et al., 2007; Hirose et al., 1991; Islaml et al., 2000; Puma-Villanueva et al., 2012; Yang and Chen, 2012). The advantage of using the CPA is the ability to generate a NN with a compact architecture for the problem to be solved. However, the main disadvantage is the decision concerning when to halt the construction process and start the pruning, and how to determine the termination condition for stopping the pruning process. Additionally, it is still not completely safe from being trapped in local minima.

4.3.4 Evolution approach

EA using randomized search strategies employ the principle of natural evolution to solve optimization problems (Ang et al., 2008; Cantú-Paz and Kamath, 2005; Oong and Isa, 2011; Yao, 1999; Yao and Liu, 1997). EA to build NNs have been introduced at three different levels to modify the NN: connection weight, NN architecture, and learning rules. Connection weight evolution targets the modification of the weighting of connections between neurons in the training phase, while the NN architecture level targets the number of neurons and/or layers, while learning rules can be modified during the learning process. All previously reported evolutionary algorithms can be considered to be population-based techniques, and evolutionary algorithms are a pow-

erful technique that can avoid local optima problems. However, it takes a long time to complete the learning process, and many parameters need to be configured in order to define the optimal case. Moreover, it is hard to determine the balance between exploration and exploitation in finding the optimal solution.

Based on the above discussion of common techniques used to generate NNs, it is clear that constructive, pruning, and constructive-pruning approaches cannot avoid problems of local optima which will prevent the design strategies from arriving at the optimal case. On the other hand, the ability of the EA to avoid local optima problems depends on the autonomous functioning of the evolutionary process, many parameters need to be manually configured, and the performance levels of this strategy are major drawbacks.

4.4 Reinforcement Learning

ANNs use function approximation in the learning process, choosing the optimal behaviour based on prior information from the training dataset. For classification problems where a training dataset is available, the function approximation technique used in ANN is very helpful in building the trained model.

For some problems where a training dataset is not available, or it is too limited, ANNs cannot reflect frequent changes in the environment, such as is the case with zero-day phishing attacks. In such cases a RL approach is suggested. RL can be defined as "learning what to do (how to map situations to actions) so as to maximize a scalar reward signal. The learner is not told which action to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying

them" (Sutton and Barto, 1998). RL uses trial-and-error to determine the optimal behaviour (Busoniu et al., 2008; Kaelbling et al., 1996; Sen and Weiss, 1999). For classification problems that apply techniques like trial-and-error, the models generated can overcome the problem of needing a training dataset that describes every situation that the system may face in the environment at execution time.

The main drawback when using techniques such as function approximation is the upper limit of the model generated by the training dataset. This means that the model cannot learn by itself how to deal with situations that are not listed in the training dataset. Function approximation can be used to design a model that can use a similar level of expertise as that of the expert involved in the training examples; however, it cannot be smarter than that expert. RL techniques can overcome this limitation, and can be used to build a model that is smarter with the ability to enhance itself. RL has an obvious drawback in that it is harder to learn the optimal behaviour, and high execution power is needed.

In the next sections terminology related to RL is discussed in more detail. A general RL problem is first described, and the Markov decision process is then clarified. subsequently, RL is then discussed and finally, how a Q-Table can be used to estimate the value functions and how generalizations can be achieved with RL is considered.

4.4.1 Reinforcement learning methods

RL is based on the ideas of dynamic programming and supervised learning, where it is used as a machine learning algorithm that simulates the human mind's way of solving problems (Schäfer, 2008). It is a method used to determine the optimal behaviour in certain environments which may be fully or partially observed (Busoniu et al., 2008).

There are many different ways to distinguish between different types of RL methods. They can be table-based or function-based methods, and could be designed as model-free or model-based methods. Table based methods store the value of each state-action combination within a table, and these methods are applied for a low-dimension discrete state space. Examples of table-based methods are Q-learning and adaptive heuristic critic. On the other hand, function-based methods learn mapping from state-action pairs to their respective values. These methods can be effectively applied for high dimensions and continuous state action pairs. Examples of the function-based methods are temporal difference (TD) and neural fitted Q-iteration.

Model-based solutions begin by building the model first based on a training dataset, then that model is used to build the controller (Doya et al., 2002). The other option does not build a model; it starts directly from the available data, which makes it faster and easier to implement (Strehl et al., 2006).

Many RL techniques have been proposed to solve different kinds of problems, and these methods can be grouped into three classes: dynamic programming, Monte Carlo methods, and temporal difference learning (Sutton and Barto, 1998). Dynamic programming techniques use strong mathematical notation, but need a model that represents the whole environment. Monte Carlo techniques do not require a model and are easier to implement, but are not suitable for incremental computation. Eventually, temporal difference (TD) algorithms do not need a model and support applications in incremental environments, but are harder to implement and analyse.

In this thesis, the proposed framework should be applicable for use with any classification problem without the need to modify any of its parameters. A NN architecture is

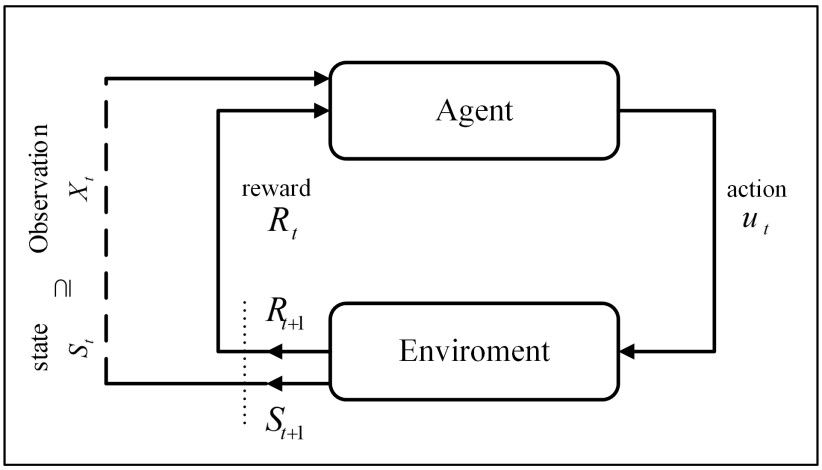


Figure 4.2: A standard RL architecture as proposed by (Sutton and Barto, 1998)

considered as the environment in which the proposed RL-Agent will apply actions. In this case, there are a limited number of actions that can be performed to make changes to the environment, as shown in the next section. Therefore, Q-learning has been chosen to implement the DENNuRL as this method is suitable for a low-dimension discrete state space.

4.4.2 Reinforcement learning problem

RL models have two main parts: the agent, and the environment. In the first step, the agent reads the state of the system S from the environment at a given time t, forming S_t. In the second step, the agent processes the environment state S_t and applies some actions u_t that cause changes in the environment. Thirdly, the changes applied by the agent to the environment produce a new state for the environment at time $t+1$ which is called $S_{(t+1)}$. The agent then computes a scalar value at time t called the reward R_t, and this reward guides the changes in the environment made by the agent, as the agent tries to maximise the reward which means that the system state is closer to the goal state as shown in Figure 4.2.

RL has the following main components (Sutton and Barto, 1998):

1. Agent: The controller of the system can, at least, observe the system's state S_t by receiving observations $X_t \subseteq S_t$. In the proposed model, the observations are represented by the NN architecture, the email record, and the NN output. The agent interacts with the system by performing actions u_t which give rewards $R_{(t+1)}$, which can be used to improve the policy. After trying these actions, a NN is generated for every action, and the reward occuring from these actions is computed and the Q-table updated.

2. Action: The action selection will influence the development of the environment. The actions u_t will target the NN architecture by changing the number of neurons in the hidden layer to reduce the Mean Square Error (MSE). There is a list of possible actions such as adding one neuron, merge two neurons, generate a new random number of neurons, and increase the number of epochs and re-train the NN. The action with the highest reward will be selected to change the environment.

3. Policy (π): This is a mapping from the state of the environment to the action to be taken in this state. The policy is the core of the RL-Agent in the sense that it alone is sufficient to determine the behaviour (mapping from the state of the environment to the action to be taken) of the RL-Agent. The RL-Agent controls changes in the environment by using the rewards computed by using Q-learning to reach the optimal NN architecture to solve a specified problem.

4. Reward/cost function: The reward function specifies the overall objective of the RL-Agent; it represents the immediate reward the agent receives for performing a

certain action in a given system state. It is used to assess the system development and guide the enhancement process, and this is accomplished by maximising the reward gained from the NN architecture.

4.4.3 Markov decision process

The mathematical basis for most theoretical RL problems is the Markov decision process (MDP), which describes a development of the fully observable controllable dynamic system (Puterman, 2014; Schäfer, 2008). The MDP is basically defined by a tuple: (S,U,T_r,R) where S is the state space which contains all possible states from the starting state until reaching the goal state. U is the action or control space, and $U(S_t)$ is the set of possible actions that can be taken when the system state is $S_t \subseteq S$ at time t. T_r is the state transition function which defines the probability of going to state $S_{(t+1)}$ when starting from state S_t when applying action U_t. Here state S_t, $S_{(t+1)} \subseteq S$ and $U_t \subseteq U(S_t)$. R is the reward function which is the reward for being in state S_t and is used to evaluate how much closer that state is to the goal state. When the next state becomes closer to the goal, the reward becomes larger and when it moves further away from the goal the reward decreases. The Markov property (Puterman, 2014; Sutton and Barto, 1998) which guides system development implies that the next state $S_{(t+1)}$ is determined only by the current state S_t and the action taken u_t at that state.

4.4.4 Q-Learning and generalization

Q-learning is an off-policy temporal difference algorithm and is recognized to be one of the simplest and most widely used technique in implementing RL. Off-policy means that the behaviour policy and the estimation policy are unrelated. The benefit of this

separation is that each policy may use a different strategy to handle actions. Q-learning is based on a table, where rows represent the system states and columns represent the set of actions which may be taken in every state. The Q-values are re-evaluated based on Equation 4.1 and stored in Q-table; these values are updated every time the system advances from state S_t to $S_{(t+1)}$. Q-learning takes advantage of TD-learning, which means it is model-free, and no information about system development is needed.

$$Q(s,a) \leftarrow Q(s_t + a_t)\alpha\,[r_{t+1} + \gamma max Q(s_{t+1},a) - Q(s_t,a_t)] \tag{4.1}$$

where α is the learning rate $0 < < 1$, which represents a trade-off between efiňĄciency and accuracy. Convergence can only be guaranteed with a decreasing value of α. Still, this is based on the assumption that the number of observations goes to inïňĄnity. The Q-learning algorithm is shown in Algorithm 4. When in use, the Q-Learning value functions are designated as a table with one cell for every state-action pair. This kind of representation is simple and straightforward, but it is restricted to problems with a limited number of states and actions. This is because, when there are a large number of states and actions, a large table will need a huge amount of memory and in some cases this becomes a limitation. This problem is known as generalization. For large problems, a combination of RL methods and a function approximation method such as NN are suggested. This is because function approximation, which is a supervised learning method depends on a training dataset and tries to generalize from it to build an approximation.

Algorithm 4 Q-learning Algorithm as identified by Sutton and Barto (Sutton and Barto, 1998)

1: Initialize $Q(s,a)$.
2: **repeat**(for each step episode)
3: Choose a from s using policy derived from Q (e.g., $\varepsilon - greedy$)
4: Take action a, observe r, s'
5: $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma max_d Q(s',a') - Q(s,a)\right]$
6: $s \leftarrow s'$
7: **until** s is terminal

In summary, a NN and RL have been chosen as the core of the proposed classification model, because the NN is one of the most powerful classification algorithms, and it gives the ability to modify the internal details of the generated model. However, the other classification algorithms discussed previously in Chapter 3 generate classification models that must be treated as a black box. The RL approach has been used to learn the optimal behaviour. It is based on the idea of trial-and-error interaction with the environment. To simplify the adaptation process for generating the best classification model, and to make it independent of the problem being solved, the NN architecture have been chosen as the environment which the RL-Agent will interact with. In addition, the number of actions that can be performed is very limited (four actions), which is the main reason for choosing Q-learning in implementing the RL-Agent. The proposed framework to generate the best NN for any classification problem is shown in the next section.

4.5 Dynamic Evolving Neural Network using Reinforcement Learning

In the proposed algorithm, a NN with three layers was chosen; these are the input, hidden, and output layers. A three-layer NN was chosen because it can solve any linear or non-linear problem (Aggarwal, 2007; Srivastava et al., 2014; Zhang and Zhang,

1999). The number of neurons in the input and output layers will depend on the number of features. The Diabetes dataset contains eight features and two output classes. Therefore, the input layer will contain eight neurons and the output layer two neurons, as we are developing a binary classification problem. The main focus in the algorithm will be to determine the right number of neurons in the hidden layer. A large number of hidden neurons may overfit the training data, and a small number will underfit the data. Overfitting and underfitting will generate a NN with a bad level of generalization, and these issues must be taken into consideration when NN is used as the core of any classification model.

During the development and enhancement of the NN, the proposed technique takes into consideration the overfitting and underfitting problems that affect NN efficiency. The overfitting problem is caused by a large number of neurons in the hidden layer. It is solved by using the merge operation. As an indication of the scale of overfitting problems, the early stopping technique (Prechelt, 2012) is used, which is accomplished by dividing the training dataset into training and validation datasets. At the time of training, the validation error is computed and if this error cannot be reduced six consecutive times, then the training process is stopped. If the validation error cannot be enhanced, this means that the architecture used has led to overfitting in the problem be solved, even if the training error is reduced. The second issue is called underfitting, and if too few hidden neurons are chosen the NN generated will not have enough power to solve the problem. This is solved by adding neurons to the hidden layer.

The steps of the DENNuRL algorithm are shown in Figure 4.3, and the following contains a full description of all of the algorithm steps.

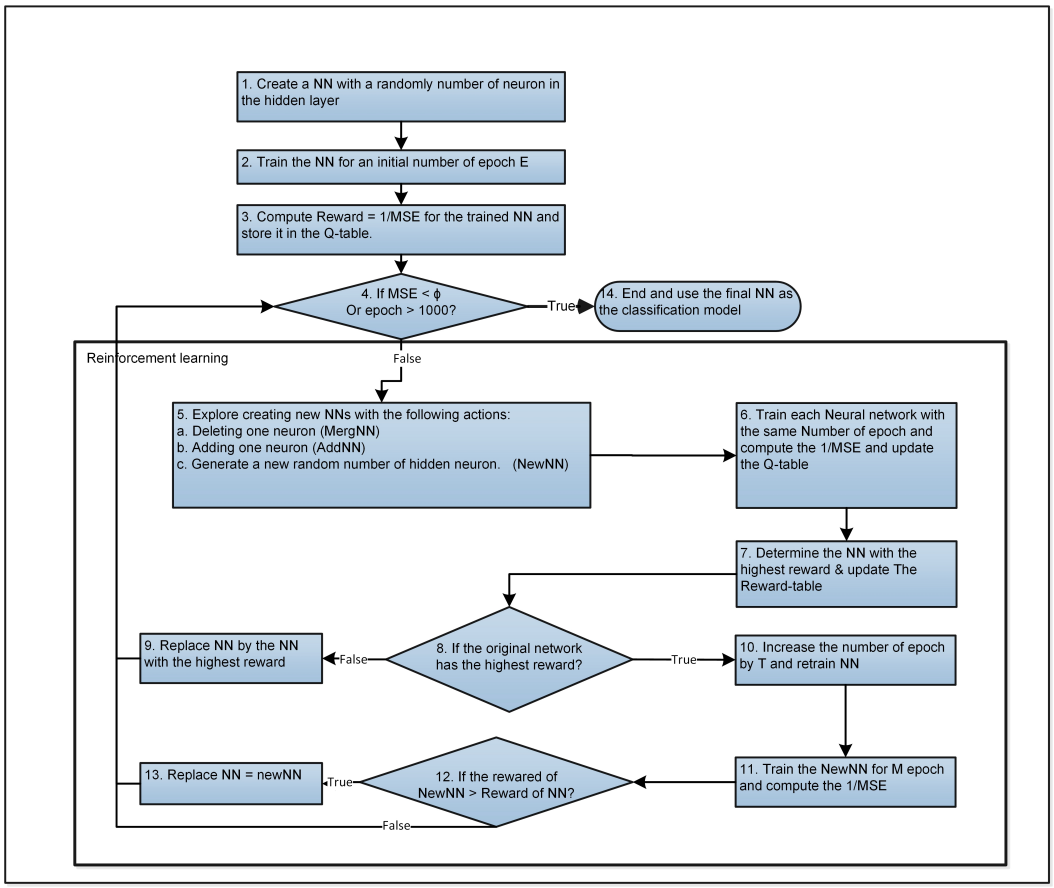


Figure 4.3: Dynamic evolving neural network using reinforcement learning (DEN-NuRL)

1. A NN is created that contains a random number of neurons in the hidden layer, and the numbers of neurons in the input and output layers will depend on the problem being solved as discussed previously.

2. The NN constructed in the first step is trained for an initial number of epochs E, and initially only a small number of epochs is chosen ($E = 10$).

3. The NN trained in step 2 is tested, and the MSE computed as shown in Equation 4.2. This determines the reward value since it has the advantage of being independent of dataset size:

$$MSE = \frac{\sum_{i=1}^{n}(o_i - t_i)^2}{n} \tag{4.2}$$

where o_i is the output for an email, t_i is the desired target for the same email, and n is the number of emails used in the evaluation process.

$$Reward = 1/MSE \tag{4.3}$$

4. Check if the termination condition is satisfied ($MSE < \Phi$) or ($epoch > 1000$) and if so go to step 14. The value of Φ is manually selected, and depends on the acceptable error rate for that problem.

5. In the proposed model, three possible actions are explored which change the system state from one state to another. The actions targeted the hidden layer in the NN, since the input and output layers depend on the problem being solved. These actions are as follows:

   (a) One neuron is deleted from the hidden layer by merging two neurons. This was proposed by Islam et al. (2009) with some modifications to simplify computation (Islam et al., 2009). To choose which neurons to merge, the following steps are applied. The correlation between hidden neurons is measured, which is calculated based on the output of the hidden neurons for the training dataset according to 4.2. The least significant hidden neuron is merged with the most correlated one, where the least significant neuron does not affect the classification process. To accomplish this action, two steps need to be applied. Firstly, the significance of every neuron is computed, which is evaluated by computing the standard deviation of the output of every neuron. A neuron's output that does not change for different inputs is considered to be less significant, as shown in Equation 4.6. Secondly,

the correlation is computed between the most and least significant neurons using Equations 4.4 and 4.5. The two neurons with the highest correlation is chosen for merging, by deleting the two correlated neurons and creating one new neuron. The input weight and bias for the new neuron is the average of the input and bias for the deleted neurons. The output weight is sum of weight for the deleted neurons.

$$R(i,j) = \frac{C(i,j)}{\sqrt{C(i,i)C(j,j)}} \tag{4.4}$$

where i, j are the two neurons whose correlation $R(i,j)$ is measured, and $C = cov(x,y)$ is the covariance between all hidden neurons which is defined as described in Equation 4.5:

$$con(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}(A_i - \mu_A) * (B_i - \mu_B) \tag{4.5}$$

where (A,B) is the output matrix of two hidden neurons for all the examples in the training dataset, μ_A is the mean of A, and μ_b is the mean of B.

$$S_d = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}|A_i - \mu|^2} \tag{4.6}$$

where S_d is the standard deviation and A is the output matrix for one neuron for all examples in the training dataset, and μ is the mean of A:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}A_i \tag{4.7}$$

(b) One neuron is added and the NN is retrained. This is done simply by adding one neuron to the hidden layer and reinitializing the NN with the new architecture. A new NN is generated with a random number of hidden

neurons; this step is added to avoid being trapped in local minima.

(c) A new random number of hidden neurons s generated and the NN is re-trained. This action has a crusual effect in avoiding being trap into local minima, where a new architechture is invistigated which is away from the local NN architecture.

6. Each of the NNs explored in step 5 is trained and the reward table is updated, where the reward from each NN is computed based on Equation 4.3.

7. RL has been used to explore the different possible actions in the modification of the NN, and the NN that returns the maximum rewards will be chosen. The Q-learning algorithm (Sutton and Barto, 1998) has been used to determine the action to be exploited using Equation 4.8:

$$Q(s,a) \leftarrow Q(s_t + a_t)\alpha\left[r_{t+1} + \gamma max Q(s_{t+1},a) - Q(s_t,a_t)\right] \tag{4.8}$$

where r_{t+1} is the received reward computed for every NN explored using Equation 4.3, α is the learning rate, γ is the discount factor, and $Q(s,a)$ is a table of s rows representing the system states and a columns representing the actions to be taken. The values of the learning rate is $0 < \alpha \leq 1$ and the discount factor $0 < \gamma \leq 1$ are manually selected by the user. In RL, how to choose the right action to be taken is crucial. This is performed by using a trial-and-error paradigm. The system tries the list of actions defined previously, and evaluates the reward from every action. The action that returns the maximum reward is taken.

8. The reward for the new NN is compared with that of the old one, and if the new NN has a higher reward, go to step 9. Otherwise go to step 10.

9. The old NN is replaced by the new one that has the higher reward.

10. The number of epochs is increased by a small amount, $T = 10$, and create a new NN which is the old NN with the new number of epochs.

11. The new NN is trained and the reward computed.

12. If the reward from the new NN is greater than the reward from the old NN, then go to step 13. Otherwise, go to step 4.

13. Replace the NN with the NN that has the highest reward.

14. The adaptation process is stopped and the final NN is used as the classification model.

4.6 Exploration Versus Exploitation

The RL-Agent is controlled by two kind of actions, which are exploration and exploitation. Exploration is the process of determining the set of possible actions that can be taken to change the system's state from one state to another. Exploitation is the process of choosing which action will be taken to modify the environment. The RL-agent learns optimal behaviour by evaluating the rewards gained when a specific action is taken. The RL-agent follows the technique of trial-and-error and tries different actions, evaluating the rewards from these actions and choosing actions that lead to maximizing the reward (Sutton and Barto, 1998). One of the difficulties that occurs in RL is the trade-off between exploration and exploitation. To choose the best action, an RL-Agent prefers actions that have been checked before and found to be effective in producing higher reward. For exploration action, the trial-and-error paradigm is

used, where the system must try different actions and progressively favour the action that returns the maximum reward. However, the RL technique can use many strategies to try to guarantee that there is a balance between exploration and exploitation. Many techniques can be used to achieve an equilibrium between exploration and exploitation, such as value-difference based exploration (VDBE) (Tokic, 2010), a greedy technique (Sutton and Barto, 1998), softmax action selection (Worthy et al., 2007), and optimistic initial values (Sutton and Barto, 1998).

4.7 Experimental Results and Discussion

To evalute the performance of DENNuRL, a well known benchmark classification problem has been used, which is the Diabetes dataset from the University of California at Irvine (UCI) machine learning repository (Bache & Lichman, 2013). The Diabetes dataset, as shown in Table 4.1 contains 768 records, which will be divided into 50% for training, 25% for validation, and 25% for testing as in the other studies that the comparison will be conducted with. The Diabetes dataset is known to be difficult to classify in the machine learning and NN field (Jayalakshmi and Santhakumaran, 2010; Perez and Rendell, 2016).

Table 4.1: Diabetes dataset description

Problem	**Number of**				
	Input attributes	**Output classes**	**Traning examples**	**Validation examples**	**Testing examples**
Diabetes	8	2	384	192	192

In addition to the evaluation meterics mentioned previously in Section 3.7.2 the Testing Error Rate (TER) and Square Error Percentage (SEP) (Prechelt et al., 1994) are used to measure the performance of the generated model. The TER is described as shown in

Equation 4.9 and it refers to the percentage of items wrongly classified by the generated NN for the testing dataset. The SEP has the advantage of being independent of the dataset size, and is described in Equation 4.10.

$$TER = \frac{100}{|P|} * \sum_{y \in p} E(y) \tag{4.9}$$

where $|P|$ is the number of examples in the testing dataset, and $E(y)$ is the error for record y, such that the error will be 1 if wrongly classified and 0 if it is correctly classified.

$$SEP = 100 * \frac{O_{max} - O_{min}}{N * P} * \sum_{p=1}^{P} \sum_{i=1}^{N} (O_{pi} - T_{pi})^2 \tag{4.10}$$

where O_{max} is the maximum output, O_{min} is the minimum output, N is the number of output nodes in the NN, N is the number of records used in the evaluation process, O_{pi} is the output of output node p for input record i, and T_{pi} is the target for record i.

4.7.1 Experimental results

An experiment was conducted to show the performance of the DENNuRL. Table 4.2, shows an example of how the NN evolved to generate the optimal NN as evaluated for the Diabetes problem. The DENNuRL algorithm keeps changing the NN architecture, as shown in numbers of neurons and epochs columns, in order to reach the best combination that solves the problem with a good level of generalization. As the MSE error guides the NN adaptation process, all of the other evaluation metrics are enhanced, which proves that the DENNuRL algorithm changes the NN architecture in the right direction. The numbers of neurons in the hidden layer are changed by exploring the different kind of actions taken to reach the goal state. In enhancement 2, a new random number is explored, in enhancement 3, one neuron is added, and

in enhancement 6, two neurons are merged. The system reaches the goal state after 8 enhancements, which gives clear evidence of the algorithm's efficiency, as discussed later in more detail.

Table 4.2: An example of NN evolving using DENNuRL

NN	ACC	Precision	Recall	F_Measure	Hidden neurons	Epoch	MSE	SEP	TER
1	76.43%	79.48%	86.00%	82.61%	22	10	15.74%	16.80%	31.3
2	76.43%	79.48%	86.00%	82.61%	8	30	15.74%	15.74%	26.96
3	77.34%	79.74%	87.40%	83.40%	9	70	15.19%	15.19%	26.09
4	77.60%	79.82%	87.80%	83.62%	9	170	14.98%	14.98%	23.87
5	78.91%	80.73%	88.80%	84.57%	12	170	14.45%	14.45%	21.74
6	78.39%	81.39%	89.00%	83.91%	11	250	14.38%	14.38%	20.87
7	79.43%	81.20%	91.00%	84.92%	19	390	14.16%	14.16%	19.13
8	81.25%	83.58%	91.50%	86.02%	19	400	13.79%	13.79%	18.26

Figure 4.4 shows the enhancement process for the NN generated by DENNuRL in terms of TER, while increasing the number of training epochs. This example shows that the best value of TER of 18.26 is produced when the number of training epochs is 400. The best TER value registered by the DENNuRL algorithm looks bad, but when compared with other studies of the same problem, the proposed algorithm in fact registers the lowest level of TER for the Diabetes dataset averaged for 50 independent runs of the algorithm as in the next section.

The accuracy of the generated NN is enhanced as the training process continues, as shown in Figure 4.5. However, in some cases, accuracy is reduced by a small amount when applying the merge operation. This is because, when the algorithm applies the merge operation, the number of neurons is reduced which will decrease the power of the model with the same numbers of training epochs. It is clearly shown that the system recovers to the previous level of accuracy as the system evolves during the development process.

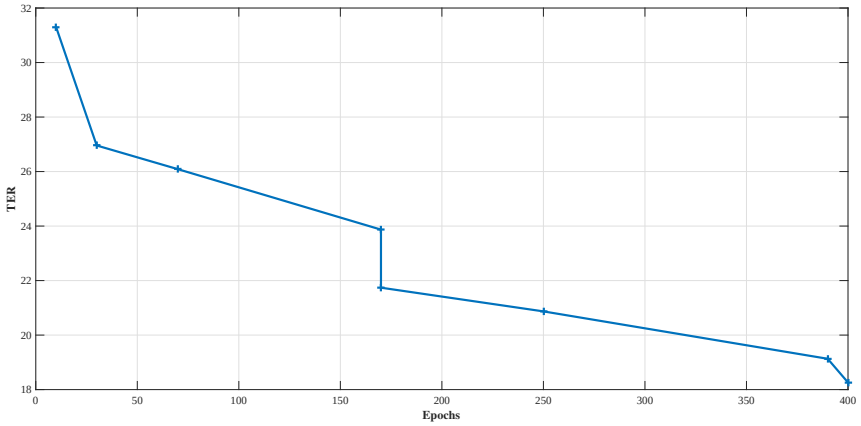


Figure 4.4: NN enhancements in terms of TER as the number of epochs is increased

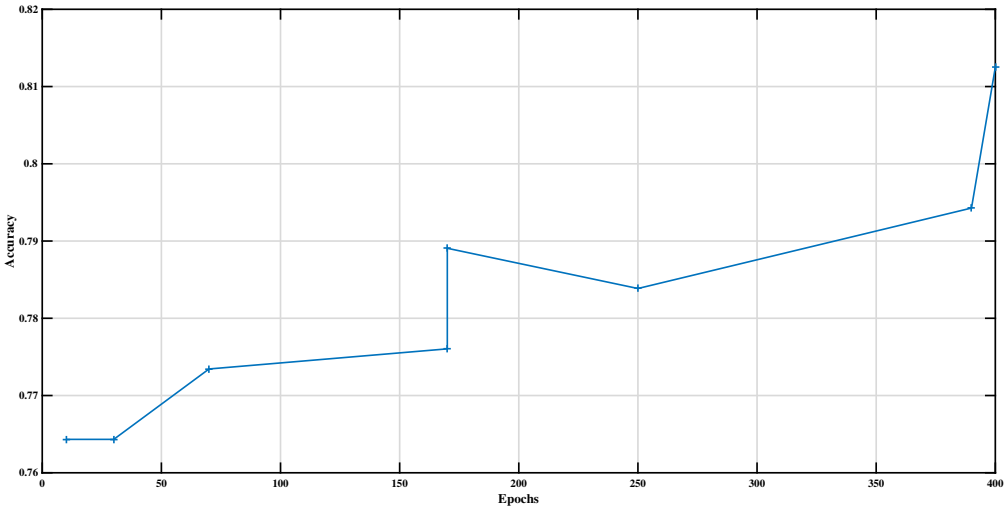


Figure 4.5: NN enhancements in term of accuracy with changes in numbers of training epochs

Table 4.3: Comparison between DENNuRL, CA, PA, and CPA. The results were averaged for 50 different run

Algorithm	TER	Epochs	Hidden neurons
DENNuRL	19.44	322.4	21.06
CA	26.04	467.5	5.96
PA	26.25	406.1	5.56
CPA	26.22	501.3	5.8
SVM	30.82	-	-
k-NN	29.06	-	-
C4.5	37.42	-	-
BP-ANN	25.87	100	-

4.7.2 Comparative analysis

Islam et al. (2009) evaluated the standard algorithms that can be used to develop the best NN architecture, (including CA, PA, and CPA) discussed in Section 4.2. He evaluated these techniques in order to compare them with his evolutionary model, which he called the Adaptive Merging and Growing Algorithm (AMGA) (Islam et al., 2009).

In this section a comparison is conducted between the proposed DENNuRL model and the standard CA, PA, and CPA methods used to generate NNs. In addition to the standard methods used to adapt NNs the comparison was conducted against modern classification benchmarks like the back-propagation NN (BP-ANN), k-NN, SVM, and C4.5. The ability of these benchmark algorithms to classify the Diabetes dataset was evaluated by Hee and Mat (2011) (Oong and Isa, 2011), and the same dataset with the same number of training, validation, and testing examples was chosen. The evaluation metrics used in the comparison are TER, epochs, and number of neurons in the hidden layer. All results were averaged for 50 different runs, each of them starting from a random initial case.

By looking at Table 4.3 it is clear that the DENNuRL algorithm outperforms other

techniques in terms of TER and the number of training epochs required to train the classification model. A lower number of training epochs mean that the proposed algorithm can reach the goal state (building the trained model) in less time, as it stops training earlier. This enhancement in the TER and the number of training epochs is due to the fact that other models always try to minimize the number of neuron in the hidden layer. These techniques are not looking for the best number of hidden neurons that can solve the classification problem with a minimum error rate, and the previous techniques fall prey to local minima problems which decrease their ability to find optimal solutions. The proposed model avoids this problem by using RL, where a set of actions is explored using a trial-and-error technique. In every enhancement, the DENNuRL tries to adapt the NN by adding or deleting one neuron, or increasing the number of epochs. In addition, to avoid local minima problems it tries to create a new NN with a random number of neurons. After trying these actions, the best one is for the next enhancement until the termination condition is reached.

To show the merits of the proposed algorithm, Table 4.4 compares the results with those of state-of-art algorithms where their models were evaluated with the same dataset. In addition to AMGA, the algorithms used in the comparison are the Variance Vullity Pruning (VNP) (Engelbrecht, 2001), Optimization Methodology for NN (OMNN) (Ludermir et al., 2006), the Hybrid Evolutionary Artificial Neural Network (HEANN) (Oong and Isa, 2011), Evolutionary Artificial Neural Network (EANN) (Oong and Isa, 2011), Self-Adaptive Growth-Brobability based Neural Network (NN-SAGP) (Ang et al., 2008), Evolutionary Programming Neural Network (EPNet) (Yao and Liu, 1997), a hybrid intelligent system that consists of the Fuzzy MinâĂŞMax neural network, the

Table 4.4: Comparison between the DENNuRL, AMGA Islam et al. (2009), VNP Engelbrecht (2001), OMNN Ludermir et al. (2006), HEANN Oong and Isa (2011), EANN Oong and Isa (2011), NN-SAGP Ang et al. (2008), EPNet Yao and Liu (1997), FMM-CART-RF Seera and Lim (2014), and FEFS-SC Luukka (2011) algorithms. The results were averaged for 50 independent runs

Algorithm	TER	Hidden neurons	Epochs	Accuracy
DENNuRL	19.44	21.06	322.4	81.25%
AMGA	21.97	4.14	390.7	-
VNP	30.9	8	-	-
OMNN	25.87	4.53	-	-
HEANN	21.33	-	-	-
EANN	21.91	-	-	-
NN-SAGP	24.16	1.4±0.69	-	75.84±2.57
EPNet	22.38	3.4	-	-
FMM-CART-RF	-	-	-	78.39%
FEFS-SC	-	-	-	75.97%

Classification and Regression Tree, and the Random Forest model (FMM-CART-RF) (Seera and Lim, 2014), and the fuzzy entropy-based feature selection combined with similarity classifier (FEFS-SC) (Luukka, 2011).

The results for DENNuRL, HEANN, EANN, FMM-CART-RF and AMGA were averaged for 50 runs starting from an initial random case, while they were averaged for 30 runs for the FMM-CART-RF and FEFS-SC algorithms. Moreover, it is not specified if the VBN and OMNN algorithms were averaged or that these were the best results achieved. The proposed algorithm gives the best performance when compared with the other algorithms, while the second best algorithm is the AMGA.

4.8 Efficiency analysis

To show the efficiency of the DENNuRL algorithm, the worst case time complexity is evaluated. The algorithm, described previously in Figure 4.3, contains two basic operation: the merge operation (step 5) and NN training (steps 6 and 7). These two basic operations are executed in the worst case 100 times, as shown in the termination

condition where the initial value for the number of epochs starts from 10 and is incremented in every iteration by 10, until it reaches the final value of 1000 epochs. To train the NN resilient back-propagation (Rprop) training algorithm is chosen because it has the following advantages (Adeoti and Osanaiye, 2013; Igel and Hüsken, 2003):

- The Rprop algorithm is a very fast and accurate training algorithm when compared with other supervised training algorithms.
- The training parameters are very intuitive and can easily be adjusted.
- The time complexity of the Rprop algorithm is $\mathcal{O}(W)$, as determined by Igel et al. (2003), where W is the numbers of connection weights in the NN (Igel and Hüsken, 2003).

For a NN with three layers that has M input neurons, N hidden neurons, and K output neurons, the number of connections between neurons that the Rprop algorithm needs to modify through the learning process is $W = N(M+K) = N(8+2)$, which $M = 8$ is the number of inputs (features), and $K = 2$ is the number of classes. The NN training is repeated for E epochs, and from the above discussion the time complexity for NN training using the Rprop training algorithm is $\mathcal{O}(E10N)$, and the number of epoch is a constant value where the final time complexity is $\mathcal{O}(N)$ (Adeoti and Osanaiye, 2013; Igel and Hüsken, 2003). The second basic operation is the merge operation, which is executed in step 5 in the DENNuRL. In this step the standard deviation is computed for every hidden neuron, as shown in Equation 4.6, and the time complexity for this operation is $\mathcal{O}(N)$, where N is the number of neurons in the hidden layer. The worst case analysis for the DENNuRL algorithm is that $Y = 100(410+N)$, where the number

of iterations (100), and number of NNs is trained (4) are relatively small. Therefore, the time complexity for the DENNuRL is $\mathcal{O}(Y) = \mathcal{O}(N)$. From the previous discussion it is clearly shown that the DENNuRL is a first-order method and the time complexity is scaled linearly with the number of neurons in the hidden layer.

4.9 Summary and Conclusion

This chapter has investigated the capability for NN adaptation using the DENNuRL algorithm. The proposed algorithm shows the ability to dynamically evolve the NN architecture to reach the optimal NN that can be used for a classification problem with a minimum error rate. RL ideas have been used to control the adaptation of the NN architecture to reach the optimal solution. Despite the fact that there are many algorithms which can be used to generate optimal NNs for classification problems, our algorithm shows promising results and a better generalisation ability (lower error rate) in building the classification model.

The proposed algorithm registers an average TER of 19.44 for 50 independent runs, which is registered better TER than those achieved in previous studies for the Diabetes dataset. Moreover, the highest average accuracy registered so far is 81.25% for the proposed algorithm. The ability of NNs is greatly dependent on their architecture (number of layers, number of neurons in the hidden layer, and connection weight).

This chapter has explained a new algorithm, the DENNuRL, which produces NNs without user intervention. The idea behind the DENNuRL is to create a NN using a modified adaptive strategy that can specify the right number of training epochs, the number of neurons in the hidden layer, and the connection weight, that can be used

in classification problems with a minimum error rate. The proposed strategy is better suited than others for these problems due to its ability to modify the NN architecture that can be used to solve classification problems with a better generalization ability. This strategy also has a lower likelihood of being trapped in architectural local optima, which is a common problem suffered by previously designed algorithms.

The proposed algorithm chooses the appropriate number of retraining epochs that is suitable for decreasing the effect of overtraining, which has a crucial effect on the generalization ability of NNs. The adaptive strategy of the DENNuRL shows that it explores the different kind of possible actions during system development. The choice of which action should be performed for NN adaptation depends on the training error associated with the designed NN. A merging operation is used in the DENNuRL to delete one hidden neuron at a time. The merge operation produces one new neuron by merging two correlated ones in such a way that a NN is produced which has the same generalisation ability as before.

The core of this merging mechanism is that it helps to remove a hidden neuron that does not affect the output of the NN, which leads to the production of more compact NN architectures. When adding hidden neurons, the DENNuRL also retrains NN to determine if the NN needs more processing power to process the training data in order to reduce the training error. Moreover, a new NN architecture is explored by creating a new NN with a random number of hidden neurons, which will allow the proposed model to overcome the problem of local optima that other solutions suffer from. RL is used in the proposed algorithm to explore the previous actions and to choose the best action that yields a better NN during the training process until the termination

condition is met.

All these techniques are adopted in the DENNuRL to design the best NN architectures with good generalisation ability. The extensive experiments reported in this chapter have been carried out to evaluate how well the DENNuRL performs on a benchmark dataset (the Diabetes dataset), and a comparison was conducted with other algorithms. The DENNuRL outperformed the other solutions for the same database with the same conditions. In its current implementation, the DENNuRL has a few user-specified parameters, although this is unusual in this field. These parameters, however, are not very sensitive to moderate changes. We can conclude from the previous discussion that the DENNuRL algorithm is a promising solution to adapt NN to solve any classification problems.

5

ONLINE PHISHING EMAIL DETECTION FRAMEWORK

5.1 Introduction

In this chapter an online Phishing Email Detection System (PEDS) is proposed. The proposed model is developed based on the ideas developed previously in Chapters 3 and 4. The intelligent model helps to identify zero-day phishing emails efficiently and dynamically. The PEDS includes a pre-processing system using Java to extract 50 features from emails. These features are reduced to an active list of features using the FEaR algorithm and is used to explore new behaviour that may exist in new datasets investigated by the PEDS. The DENNuRL algorithm use the active list of features exploited by the FEaR algorithm to build the detection model; this model uses a NN as the core of the model.

5.2 Zero-Day Phishing Attack

One of the primary goals of this thesis is to detect zero-day attacks of phishing email. Current methods do a reasonable job of identifying known phishing emails, but there is a time delay before those newly identified phishing sites are added to blacklists. A user who visits a new phishing site is vulnerable until that site is added to blacklists. Using heuristics such as the FEaR, DENNuRL, PEDS, and RL-Agent algorithms will provide the ability to detect new phishing emails.

Detecting zero-day phishing attacks has a crucial effect on online transactions, but only a limited number of studies have designed methods to handle such attacks. Any model that is supposed to detect zero-day phishing attacks needs to have the ability to dynamically adapt the detection model to reflect changes found in new phishing emails. In addition, it should have the ability to explore new behaviours in newly received email in the online mode. None of the previous studies give a clear idea of how to explore these new behaviours in zero-day phishing emails.

Any detection model that is designed to handle zero-day phishing attacks should have the following properties:

- The model should give a low FPR, which means that a legitimate email should not be identified as a phishing email. This is the most important variable in the field of phishing email detection, because the legitimate email classified as phishing may contain important information for the user, and it may cause a loss of important information.

- The model should provide a high TPR and therefore detect the majority of

phishing emails.

- With a high TNR, the model should identify legitimate sites and correctly verify that the email is legitimate.

5.3 Proposed Framework for Zero-Day Phishing Attack Detection

The proposed model called PEDS has the ability to explore new behaviours in any new dataset using a novel algorithm called the FEaR. The PEDS will incrementally enhance itself to handle new attacks depending on the idea of RL. A NN is used as the core of the detection model and a novel algorithm called the DENNuRL is proposed to provide the best NN that can be used to solve the problem at hand. Moreover, the detection model is automatically changed to reflect changes in zero-day phishing attacks.

The PEDS is an online phishing email detection method based on supervised and unsupervised machine learning techniques. The supervised machine learning technique uses a training dataset to build the detection model, while unsupervised machine learning tries to adapt the detection model using emails newly delivered to the system. The proposed model combines NN, RL, data mining associative classification techniques and a set of algorithms to detect phishing attacks.

5.3.1 Online system model

The proposed online PEDS, as shown in Figure 5.1, starts by extracting fifty features from the offline dataset, and then the FEaR algorithm described in Section 3.5 is applied. Applying the FEaR algorithm involves the following benefits. Firstly, it will

discover phishing behaviours used in the offline dataset. Secondly, it will decrease the complexity of the generated model by minimizing the number of input neurons and the number of connections between the input and hidden layers. Thirdly, it will speed up the adaptation process to generate the best NN, and the speed of adaptation will have a crucial effect on the performance of the online system. Finally, it will accelerate the classification process. In the next step, the DENNuRL, as shown in Section 4, will be used to generate the PEDS which will be utilized in the online mode to classify emails. Later the RL-Agent will be used to continuously adapt the PEDS to reflect newly explored behaviour in the online mode.

5.3.2 RL-Agent algorithm

After producing the first PEDS using the DENNuRL algorithm based on the training dataset in the offline mode, the system starts the development process by reading unclassified email in the online mode. Figure 5.2 shows the RL-Agent algorithm, the system classifies emails into phishing and ham email.

The RL-agent keeps watching the output of the PEDS in the online mode and works as follows:

1. Emails are classified one at a time and passed to the evaluation system as shown in Figure 5.2, and are then passed to step 2.

2. If an email is classified with a very high accuracy $Accuracy \geq \alpha$ and $Accuracy \leq \beta$, go to step 3. Otherwise the next email is read,returning to step 1.

3. Email classified using the PEDS are collected in a new dataset (NewDataset), and all features originally extracted in the pre-processing phase are considered

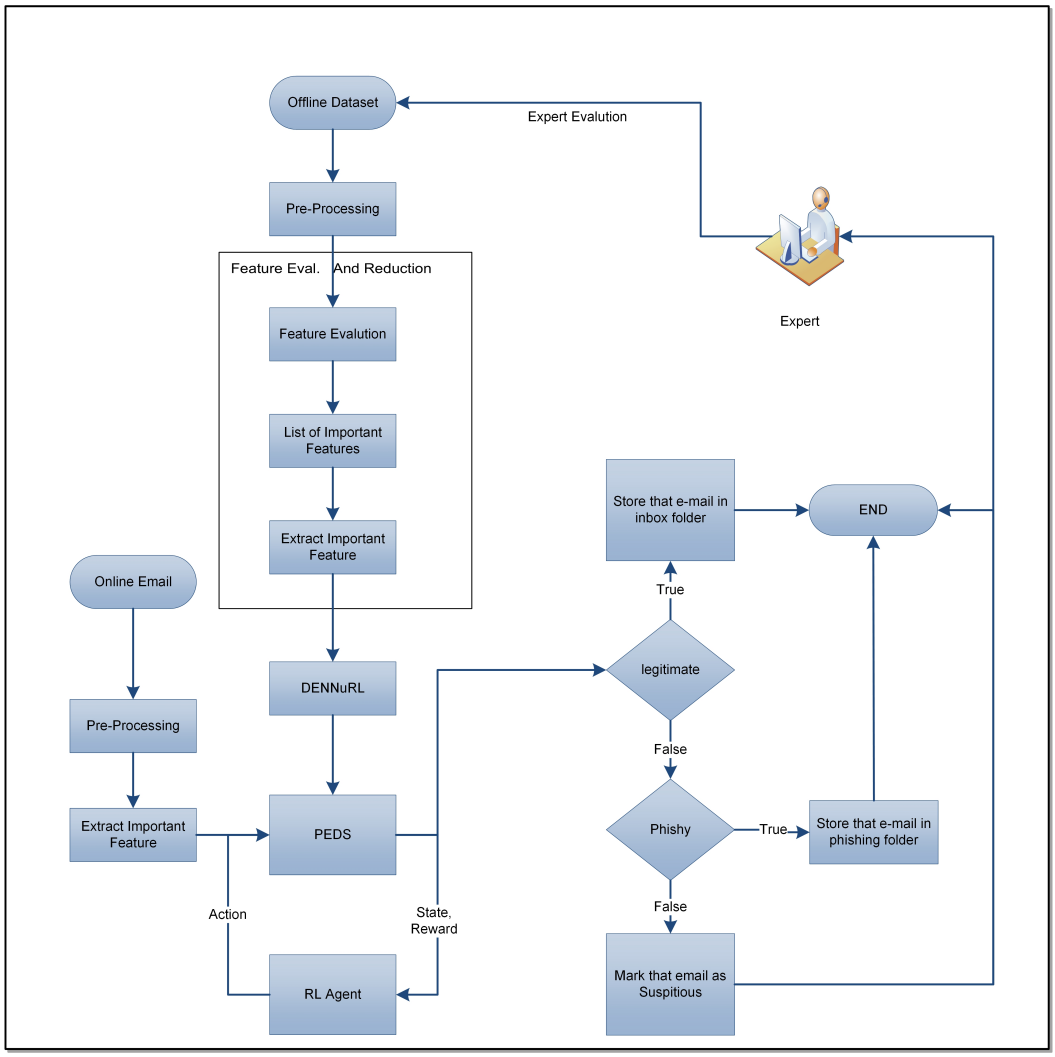


Figure 5.1: Online phishing email detection system

in this step.

4. If the number of emails collected in the $NewDataset = \delta$, go to step 5. Otherwise the next email is read, returing to step 1. The value of δ is dynamically changes which is 10% of the upToDateDataset, which is initially equal to the offline dataset.

5. The system merges the NewDataset with the upToDateDataset, and the result is stored in the NewDataset.

6. The FEaR algorithm is applied to the NewDataset to explore new phishing

behaviours in the NewDataset.

7. The list of important features is determined.

8. The list of important features from the NewDataset is extracted.

9. A new PEDS is generated using the DENNuRL algorithm to reflect changes in the dataset.

10. The newly generated PEDS is tested for the reference dataset (offline dataset) and the upToDateDataset.

11. If the new PEDS is better than the previous PEDS, then replaces the PEDS with the new one, the upToDateDataset is replaced with the NewDataset and the list of important features is updated. Elseif the reference PEDS (the first PEDS generated for the offline dataset) is better than the new one over the reference dataset the reinitialize the hole system. Else, the adaptation process is ignored and the previous PEDS is used.

The proposed technique to adapt the PEDS has a limited number of user-defined parameters (α, β, δ), which is not usual in this field. The values of these parameters are manually configured after conducting a preliminary test run, and they are not intended to be the best possible values. However, the values of these parameters are not sensitive to moderate changes in the environment. In addition to the RL-Agent used to control the behaviour of the DENNuRL described previously in Section 4.4, the following section describes the main part of the RL-Agent used to control the adaptation of the PEDS as shown in Figure 5.2:

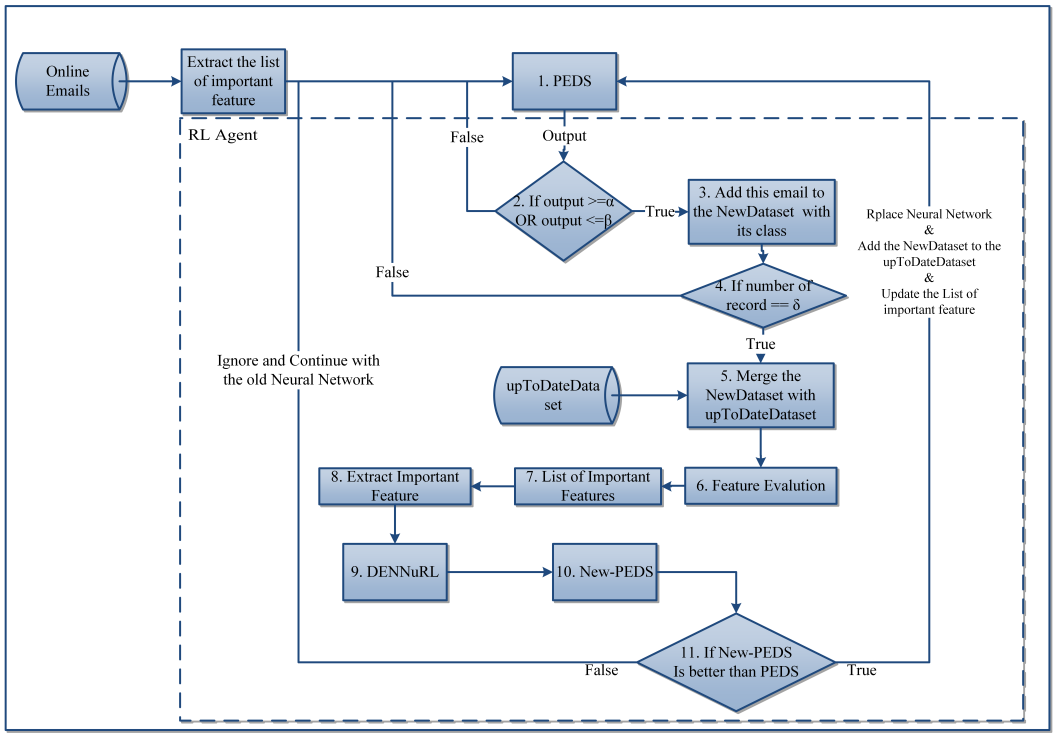


Figure 5.2: Reinforcement learning agent

1. Agent: In the proposed model, the observation will be the output of the PEDS, and this is the new email detected as phishing or legitimate email with high accuracy. These email will be collected in the newDataset, when the number of email in the newDataset is 10% of the upToDateDataset merge these two dataset in the newDataset.

2. Action: After building the newDataset, the system will explore any new behaviour which may exist in the newDataset by using the FEaR algorithm, and then the DENNuRL algorithm is applied to design the new phishing email detection system. At this stage, the RL-Agent will compare the new detection system with the old one. If the new detection system gives better results, it replaces the old PEDS, the upToDateDataset dataset is updated, and the list of important features is updated. Otherwise, the adaptation process is ignored.

3. Policy (π): This is a mapping from the state of the environment to the action to be taken in this state. The policy is the core of the RL-Agent in the sense that it alone is sufficient to determine the behaviour of mapping from the state of the environment to the action to be taken by the RL-Agent.

4. Reward/cost function: The reward function specifies the overall objective of the RL-Agent. The reward will depend on the ouput of the PEDS, where the system with lower MSE will be better.

To handle the possibility of reinforcing a wrong result, the proposed model tests the new PEDS for a reference dataset which is the offline dataset that is used to generate the first detection model. If the new PEDS cannot achieve better performance compared with the first PEDS, then we assume that the system is reinforcing a wrong classification. In such a case the enhancement is ignored and the system is reinitialised.

After all there is a possibility of miss-classification for some email that the model is not trained on (zero-day attacks); the proposed model try to solve this problem using exploration-exploitation mechanism as shown in step 2 in the RL-Agent Algorithm. The RL-Agent Algorithm explores adding a group of emails (10% of the offline dataset) to the NewDataset. These emails are classified by the current PEDS with high accuracy ($\geq 95\%$). The NewDataset will contain the old dataset plus the new group of email.

The RL-Agent will generate (exploit) a new PEDS and compare it with the previous one. In this case, the new PEDS will advance one step in every enhancement towards the new kind of attacks (zero-day attack). Since the email is classified by the previous PEDS with 95% accuracy as phishing will be classified by the new PEDS with 100% accuracy. Furthermore, the expert can enhance the system ability to handle a new

kind of attacks by manually classifying suspicious emails and adding any new available dataset to the offline dataset. Lastly, as we discussed previously, the new PEDS will always be experimented with the reference dataset and the upToDateDataset to make sure that the new model does not reinforce the wrong results.

5.4 Experimental Results and Discussion

5.4.1 Dataset

The experiments were conducted using a dataset combined from three publicly available datasets which were described previously in Section 3.7.1.

To train and test the PEDS, the dataset has been divided into offline and online datasets. The online dataset was assumed to be unclassified, and the total number of emails used was 4951 for ham emails and 7315 for phishing emails. From the phishing emails in every experiment, 4951 were chosen randomly. The total number of emails was 9902, and 4000 were selected randomly as an offline dataset. The rest of the emails were used as an online dataset to assess the system's development in response to a zero-day phishing attack. The experiment was conducted and repeated fifty times, and in every iteration the offline dataset was chosen in a random order and divided into 70% for training, 15% for validation and 15% for testing. The results were recorded for every iteration and then the average was computed for all iterations to ensure the reliable performance of the system.

5.4.2 Evaluation metrics

The set of evolution metrics described previously in Section 3.7.2 were used to evaluate the performance of the proposed model.

5.4.3 DENNuRL

As discussed before, the DENNuRL algorithm, as shown in Figure 5.1, is used to generate the first PEDS employed in the online mode. Later in the online mode, the same algorithm is utilized as part of the RL-Agent as shown in Figure 5.2. In both cases, it will be used to automatically choose the best NN to solve the problem at hand.

Table 5.1: An example of the NN adaptation using the DENNuRL algorithm

NN No.	Number of neuron in the hidden layer	Number of epoch	MSE error
1	27	10	0.024355
2	37	10	0.023192
3	38	10	0.023105
4	39	10	0.023026
5	6	10	0.022724
6	7	10	0.022211
7	7	60	0.012845
8	8	60	0.01118
9	8	360	0.010688
10	20	610	0.010609
11	19	610	0.010607
12	19	1010	0.009976

The first PEDS is built by using the DENNuRL algorithm based on the offline dataset of 4000 emails. As an example of the usage of the DENNuRL, Table 5.1 shows the enhancement of the NN from the NN randomly generated in the first step to the final version to be used as a PEDS. The level of MSE error guides the enhancement process, where the NN with the lowest MSE error will be better. The benefits from these small changes in MSE error will be clearly seen when the rest of the evaluation metrics are taken into consideration as shown in the next section.

Table 5.1 and Figure 5.3 shows the enhancement performed by the DENNuRL algorithm in terms of MSE error, according to the system's development. MSE error is

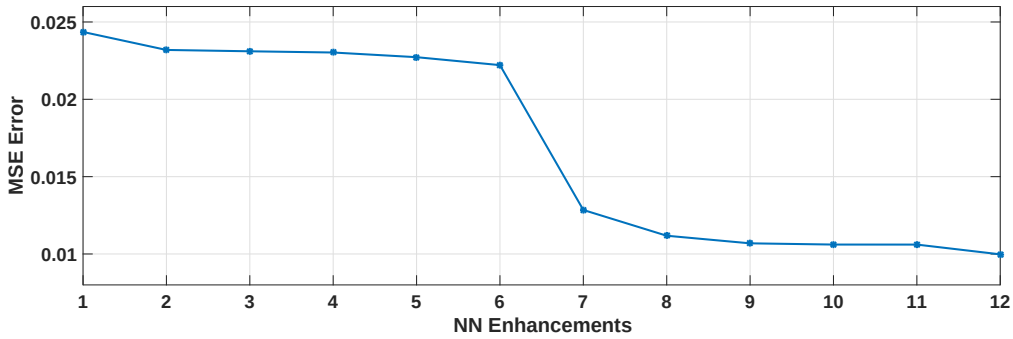


Figure 5.3: An example of NN enhancement using the DENNuRL in terms of MSE error

computed based on Equation 4.2 as described previously in Chapter 4. In the first NN shown in Table 5.1, the number of 27 neurons is selected randomly by the algorithm, and then the system tries the different actions and selects the NN network that returns the maximum rewards (minimum MSE error). In the second enhancement, a new random NN is selected, and at the third enhancement one more neuron is added to the hidden layer. At step 11, two neurons are merged in the hidden layer, and finally in the last enhancement the training epoch is increased in order to reduce the MSE error.

5.4.4 Online phishing email detection system

After building the first PEDS using the DENNuRL algorithm, the system start classifying the online dataset described previously in section 3.7.1 while PEDS classifies email in the online mode the RL-agent keep track of the output of the system. To compare the ability of the PEDS to detect phishing emails for the system before and after the adaptation, a Detection Error Trade-off (DET) curve (Adler and Schuckers, 2005; Martin et al., 1997) is used. The DET curve is a graphical representation that shows a trade-off between the two types of error: missed detections (y-axis) and false alarms (x-axis). Missed detection (false negative) is where a phishing email is clas-

sified as legitimate, and a false alarm (false positive) is where a legitimate email is classified as phishing. The two kinds of error were evaluated for the two FEDS for the same dataset selected randomly from the original dataset. The first system is the PEDS developed initially before applying the adaptation process, and the second is the PEDS after applying the adaptation using the RL-Agent. The DET gives a good indication of the efficiency of the system at different operation points. The DET curve shown in Figure 5.4 clearly indicates that the adapted system has lower probabilities of false rejection and false acceptance rates at all operation points.

The DET curve is generated by sweeping the decisions threshold over the range of scores produced by the detection system, and performance is evaluated at small, successive intervals (Yang et al., 2000). Then the normal deviations of the miss versus false alarm rates at each interval are plotted. The detection error trade-off function, which takes in a list of prediction values and a list of truth values, produces output containing the false alarm (FPR) and missed detection (FNR) for all threshold values. The DET curve, which is the line showing the trade-off between FPR and FNR, is typically viewed in logarithmic coordinates. The DET curve shown in Figure 5.4 shows that the two kinds of error, false alarms (x-axis) and miss detection (y-axis) are related. The nature of this relationship is determined using Equation 5.1 (Auckenthaler et al., 2000):

$$y = -(\frac{1}{\sigma_T}x + \frac{\mu_T}{\sigma_T}) \tag{5.1}$$

where (μ_T, σ_T) are the parameters of the target distribution, which is assumed to be Gaussian. A special point in the DET curve, where the probability of false alarms is equal to the probability of missed detection of (the targeted email) is called the Equal

Error Rate (EER) as shown in Figure 5.4. The EER can be evaluated using Equation 5.2.

$$x = y = -\frac{\mu_T}{1+\sigma_T} \quad \text{and} \quad P_{EER} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt \tag{5.2}$$

By visual inspection of the DET curve, we can confirm that the PEDS after adaptation is better taking into consideration the two kinds of error. A weighted average of missed detection and false alarm rates may be used as a kind of figure of merit or cost function. The point on the DET curve where such an average is minimized is called the EER. In Figure 5.4 these points are indicated by arrows. Table 5.2 summarizes the enhancement

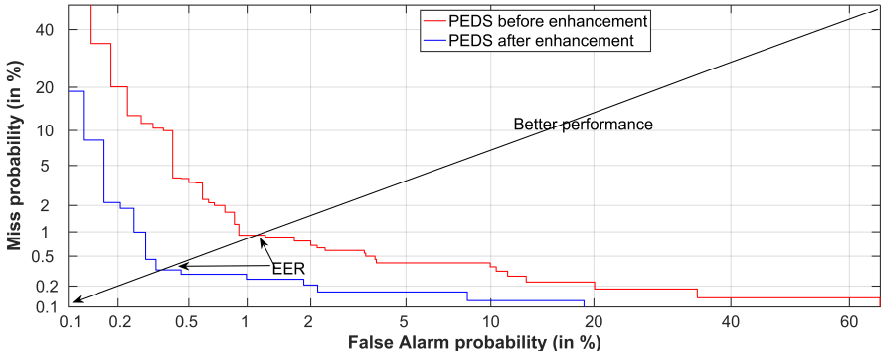


Figure 5.4: Detection error tradeoff (DET) for PEDS before and after adaptation

of the PEDS. To show how the system evolves in the online mode, eight different metrics were computed, which were accuracy, precision, recall, F_Measure, TPR, FPR, TNR, and FNR after every adaptation of the FEDS in the online mode. Figure 5.5 show how the system evolves in terms of accuracy, where fifteen enhancements have been conducted throughout the processing of the online dataset of 5902 emails, and the results clearly show an enhancement in all metrics over time. Finally, to show the reliable performance of the proposed model, the experiment was repeated fifty times. Every experiment started from a random case and the final PEDS was produced after the enhancements were tested. Table 5.3 shows a summary of the different metrics

Table 5.2: PEDS enhancements in terms of FNR, FPR, TPR, Accuracy, Precision, Recall, and F_Measure

Enh.	FNR	FPR	TPR	TNR	ACC	Precision	Recall	F_Measure
1	1.89%	5.15%	98.11%	94.85%	96.49%	95.06%	98.11%	96.56%
2	1.89%	4.82%	98.11%	95.18%	96.65%	95.36%	98.11%	96.72%
3	1.52%	5.15%	98.48%	94.85%	96.67%	95.08%	98.48%	96.75%
4	1.52%	5.15%	98.48%	94.85%	96.67%	95.08%	98.48%	96.75%
5	1.52%	5.11%	98.48%	94.89%	96.69%	95.11%	98.48%	96.77%
6	1.56%	5.11%	98.44%	94.89%	96.67%	95.11%	98.44%	96.75%
7	1.60%	4.36%	98.40%	95.64%	97.02%	95.79%	98.40%	97.08%
8	1.60%	4.48%	98.40%	95.52%	96.96%	95.68%	98.40%	97.02%
9	1.48%	3.57%	98.52%	96.43%	97.48%	96.53%	98.52%	97.52%
10	1.07%	2.82%	98.93%	97.18%	98.06%	97.25%	98.93%	98.08%
11	1.07%	2.82%	98.93%	97.18%	98.06%	97.25%	98.93%	98.08%
12	1.11%	2.82%	98.89%	97.18%	98.04%	97.25%	98.89%	98.06%
13	0.99%	2.62%	99.01%	97.38%	98.20%	97.45%	99.01%	98.22%
14	1.03%	2.03%	98.97%	97.97%	98.47%	98.00%	98.97%	98.49%
15	1.15%	1.74%	98.85%	98.26%	98.55%	98.28%	98.85%	98.56%

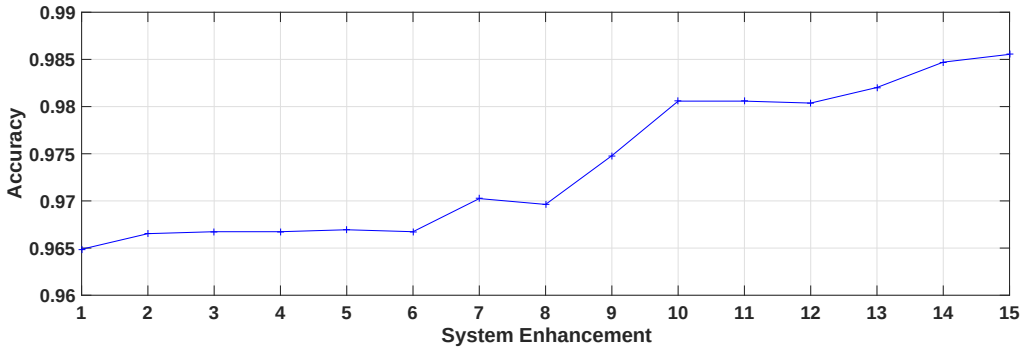


Figure 5.5: System enhancements in terms of accuracy

Table 5.3: Results for the online PEDS averaged over 50 independent runs

FNR	FPR	TPR	TNR	ACC	Precision	Recall	F_Measure	AUC
0.93%	1.81%	99.07%	98.19%	98.63%	98.21%	99.07%	98.64%	99.43%

used in the assessment of PEDS performance; the experiment was repeated fifty times and an average is taken.

5.4.5 Comparative analysis

Table 5.4 shows the outcome of the comparison of the present results with those of previous work. Islam and Abawajy (2013) proposed a multi-tier classification model that used a combination of three classification algorithms (SVM, AdaBoost, and Naive-

Bayes) (Islam and Abawajy, 2013). In that study, twenty-one features were extracted from email header and content. The highest accuracy registered was 97% with the arrangement ($C1-C3-C2$) where C1 was the support vector machine, C2 as Boosting (AdaBoost) and C3 as Bayesian (Naive Bayes). However, the proposed algorithm could not be modified to handle new types of attacks and the classifier combination did not have the ability to be adapted to reflect changes in online emails. Almomani et al. (2013b) proposed a novel model to handle zero-day phishing attacks (Almomani et al., 2013b). Twenty-one features were used in building the detection model and 98% accuracy was achieved, but the proposed model was tested in the online mode with only a tiny dataset of 300 phishing emails and 2000 legitimate emails.

Ramanathan and Wechsler (2012) used the PLSA topic trainer to build their trained model, and a high accuracy was achieved at 97.7% while using the largest dataset in comparison with other studies (Ramanathan and Wechsler, 2012). The main disadvantages of this study included ignoring many features related to URLs, which are the main connections between emails and phishing websites. Furthermore, in the selection of the dataset it was assumed that the Enron email dataset was free from spam and phishing email, which has not been proven by the dataset producer. Finally, they did not consider redundancy in this vast dataset.

The offline FEDS proposed previously in Chapter 3 was found to be better than previously proposed phishing email detection systems (for example: Khonji et al. (2012), Gansterer and PÃűlz (2009), Chandrasekaran et al. (2006), Ma et al. (2009), Abu-Nimeh et al. (2007), Toolan and Carthy (2010), and Hamid & Abawajy (2011)), and a full discussion of these techniques is explained in Section 3.7.7. The main disadvan-

Table 5.4: Comparison of our approach with previous work

Author	No. of features	Algorithm used	Feature approach	Sample	Results
Islam and Abawajy (2013)	21	SVM, AdaBoost, and NaiveBayes	Hybrid	Undetermined size, but the same source as the proposed approach	ACC 97% FP 2% FN 9%
Almomani et al. (2013b)	21	ECM & DENFIS	Hybrid	Phishing 4300 Legitimate 6000	ACC 98%
Ramanathan and Wechsler (2012)	200 topics	AdaBoost	Content	400,000 email	ACC 97.7%
Offline PEDS proposed in Chapter 3	33	Random Forest	Hybrid	Phishing 4559 Legitimate 4559	ACC 97.98%
Online PEDS	50	Neural Network & Reinforcement Learning	Hybrid	Phishing 4559 Legitimate 4559	ACC 98.6%

tage of the offline PEDS developed using The Random Forest classification algorithm is that it does not have the ability to adapt to reflect changes in the environment. Furthermore, the proposed online PEDS outperforms the offline PEDS in terms of all metrics used to evaluate the proposed model.

The present online PEDS approach uses fifty hybrid features which are reduced to a lower number of effective features by using the FEaR algorithm depending on the new dataset explored in the system developments. The number of important features changes dynamically to reflect changes in the dataset. The dataset grows dynamically by running the proposed model in the online mode. The PEDS is dynamically adapted

to reflect changes in the online environment using the RL-Agent. Only a few studies have so far considered the handling of zero-day phishing attacks, and yet our model has been proven to have the capability to handle these attacks with high accuracy, Table 5.3 shows a summary of the evaluation metrics used to evaluate the proposed model averaged over 50 independent runs.

5.4.6 Implementation and execution time

The proposed model was tested on Intel(R) Core(TM) i7-3537U CPU @ 2.00GHz (4 CPUs), with 8 GB RAM and Window 10 operating System. The pre-processing system was implemented using JAVA programming language (NetBeans 8.0 IDE). The PEDS, DENNuRL, and RL-Agent were implemented using MATLAB R2015a. In addition to the computation complexity shown in the next section, Table 5.5 shows the actual time needed to execute the different parts of the proposed model. The long time needed to execute the RL-Agent does not affect the overall system performance since this step is done in parallel with the classification process. The RL-Agent observe the output of the PEDS and tries to develop a new one. The current PEDS is replaced by a new one when the RL-Agent finish the enhancement process.

Table 5.5: The proposed system execution time

Program Component	**Execution Time**
Pre-processing	Average of 3.11 Seconds for each email.
PEDS	Average of $7.2 * 10^{-3}$ seconds for each email.
RL-Agent	Average of $2.5 * 10^{3}$ seconds to develop an enhanced PEDS.

5.5 Efficiency Analysis

In this section, the time complexity for the operations of the online PEDS is evaluated. PEDS is generated from the offline dataset only once, the first time the adaptation process is applied when classifying the online dataset. Emails in the online mode are fed into the system one at a time, and the preprocessing system extracts the list of features from each email and sends it to the PEDS for classification. Then the output of the classification is sent for evaluation to determine the right action corresponding to that email. Also, the RL-Agent takes the output of classification to adapt the PEDS, as shown previously in Figure 5.1. Since the first PEDS generated the first time is executed only once, it will be excluded from the analysis. Then each email will be subject to three phases: pre-processing and FEaR, classification using PEDS, and RL-Agent.

The first phase is relatively light weight where each feature is extracted directly from the email header or content, which means that it could be accomplished in linear time $\mathcal{O}(M)$, where M is the number of emails. In the second phase, email classification is conducted using the NN, which uses the value of features computed in the first phase as input to the input layer and the output for every neuron is computed using Equation 5.3.

$$Y = \sum_{i=1}^{n} x_i w_i - \Theta \tag{5.3}$$

where n is the number of neurons in the previous layer, Θ is the bias applied to that neuron, and $x_i w_i$ is the input and connection weight for that input. The result is passed to the activation function to bound the result between $[0 \leq Y^{sigmoid} \leq 1]$. The

activation function used in this case is the sigmoid activation function as shown below:

$$Y^{sigmoid} = \frac{1}{1+e^{-Y}} \tag{5.4}$$

The output for the neurons at the output layer is computed using Equation 5.3 for every email. The time complexity for the NN is $\mathcal{O}(N)$, where N is the number of neurons in the NN.

In phase three, the main operation executed in the RL-Agent is the call for the DENNuRL algorithm, and the complexity of this algorithm has been evaluated previously in Section 4.8. The time complexity for DENNuRL is scaled linearly with the number of neuron in the hidden layer $\mathcal{O}(N)$.

Based on the above discussion, the time complexity for the online phishing email detection system is $\mathcal{O}(M*(N+N)) = \mathcal{O}(M*N)$. The main conclusion is that the adaptation mechanism used in this chapter provides linear adaptation capability.

5.6 Summary and Conclusion

Phishing is one of the most serious cybercrime threats that reduces customer trust in e-commerce. This research shows how a NN with RL can be used to build a powerful model to detect zero-day phishing attacks with high performance levels and acceptable error rates. A novel algorithm has been proposed for developing a classification model based on a NN, as well as an algorithm to explore new behaviour and adapt the phishing email detection system dynamically, with the capability to explore new behaviours. The significance of the proposed model lies in following respects. Firstly, the PEDS is capable of online detection with the ability to adapt itself to reflect changes in the

environment. Secondly, the FEaR algorithm is used to determine the list of important features in a dynamic environment. Thirdly, the RL methodology which has been used for the first time in this field, used in the proposed approach to increase the ability of the system to evolve based on changes in the environment. Fourthly, a NN is used as the core of the detection model, and the RL-agent is used to adapt the NN to build the best possible NN for use in the detection model. Finally, the performance of the online PEDS was evaluated using a set of metrics and compared with previously proposed approaches for phishing email detection. The proposed approach registers high accuracy, TPR, and TNR at 98.63%, 99.07%, and 98.19% respectively. In addition, it shows low FPR and FNR, at 1.81% and 0.93% respectively, and the rest of the metrics used to evaluate the proposed approach are summarized in Table 5.3.

6

CONCLUSION AND FUTURE WORK

This thesis focuses on developing a novel online phishing email detection system. This chapter gives a short look back on what has been discussed so far. Considering the research questions related to the aim and objectives of the study and whether or not they are having been answered as well as suggesting how research in this area could perhaps develop in the future. This concluding chapter introduces the research contributions and assumptions, accomplishments and limitations. Finally, recommendations are made for future work.

6.1 Summary of This Thesis

The problem of phishing attacks is growing in scale and complexity, since phishers use new zero-day phishing attacks and continuously adapt their strategies to lure victims. The following discussion summarizes the thesis chapters and in the next section a brief discussion is given of its accomplishments and contributions.

In Chapter 2, the phishing problem was discussed to lay the basis for the subsequent chapters. What exactly is phishing? What is its history and the current situation and how do previously proposed solutions cope with these attacks? From the different levels where phishing attacks are launched, phishing attacks at email level are the subject of the proposed model, as this is a bottleneck where most phishing attacks start from, and where changes can be most easily made and tested. Moreover, a more secure environment can be produced if phishing attacks are detected at an earlier stage such as at email level. Afterwards this chapter discussed a number of previous studies about the phishing problem itself, the reasons why current systems fail and why users are blinded by the attacks.

Chapter 3 investigated the list of features that represent the different aspects of emails which include email headers and content. The list of features includes features described in previous studies and seven more are newly proposed in this thesis. One of these newly proposed features is dynamically updated every sixty minutes from the PhishTank blacklisted URLs. Moreover, a new pre-processing program has been proposed to extract the list of features using the JAVA programming language so as to be platform-independent. To explore the active list of features that accurately represents the phishing email in an offline dataset, a new algorithm is proposed which is called

the FEaR algorithm. Finally, to evaluate the ability of the pre-processing system and the FEaR algorithm, an experiment was performed using 10 different classifiers used previously in the literature.

Before presenting the main phishing email detection system in Chapter 5, Chapter 4 presents a new algorithm to build a dynamically evolving neural network that can be used to generate a classification model. A benchmark dataset (the Diabetes dataset) is used to test the ability of DENNuRL algorithm to generate the best NN as a classification model and to compare the results with those of state of the art algorithms used previously for the same purpose. The results of the comparison (see Section 3.7.5) prove that the DENNuRL algorithm can generate a better classification model than previous techniques.

Finally, Chapter 5 shows the final phishing email detection model, the PEDS. This model encompasses firstly, a pre-processing algorithm that extracts fifty features from the main email components. Secondly, the FEaR algorithm determines the list of most relevant features in the offline dataset and it explores any new important features from zero-day attacks introduced in the up-to-date dataset collected by the RL-Agent. Thirdly, the DENNuRL is responsible for generating the first phishing email detection system based on the offline dataset, and later it is used to adapt the model to reflect any changes in the environment. Finally, the reinforcement learning agent (RL-Agent) is used to control the environment by exploring and adding new email data to the up-to-date dataset and updating the phishing email detection system to reflect these changes.

6.2 Research Contributions

Many contributions have emerged from this research investigation which can be very helpful for all researchers engaged in the field of machine learning and phishing detection using the NN and RL. A summary of the study's main contributions is as follows:

- Fifty features and patterns represent different aspects of emails; these features are extracted from three different sources which are email content, header, and external sources. The selected list of features is chosen based on previous studies, while seven of them are newly proposed in this thesis. The extraction process is divided into three different layers to simplify the process, depending on the feature types and sources. The proposed pre-processing algorithm was evaluated using ten classification algorithms used previously in the literature. The offline detection model was compared with a set of previous studies as shown in Table 3.9, and it shows better results in terms of all metrics considered. The highest accuracy registered was 97.98% for the Random Forest algorithm, and the rest of the evaluation metrics are summarized in Table 3.8. The results of the offline detection model show that the proposed pre-processing algorithm is better than other strategies, where the same classification algorithm and the same dataset were used in the evaluation.

- A feature evaluation and reduction (FEaR) algorithm has been proposed for two purposes. Firstly, it has been applied to reduce the number of selected features to give a list of features which is the most important and capable of

detecting the desired class of objects with an acceptable error rate. Secondly, in the online PEDS it is responsible for the exploration of new behaviour that may be available in the up-to-date dataset constructed by the system. Table 3.6 shows that the FEaR algorithm can discover different behaviours when the dataset used to explore these behaviours is changed. Furthermore, the FEaR algorithm can be used to dynamically rank the list of selected features to determine their importance in the selection process, as shown in Table 3.7.

- A dynamic evolving neural network using reinforcement learning (DENNuRL) algorithm has been proposed. This algorithm is the core of the phishing email detection system and is responsible for building and dynamically adapting the classification model based on the NN and RL. The DENNuRL algorithm has been tested with a benchmark dataset (Diabetes dataset) and it was proven that it can generate a classification model with good generalization (low testing error rate) when compared with other algorithms that build classification models using NNs. The proposed algorithm registered a value of TER of 19.44, which is the lowest testing error rate registered for the Diabetes dataset, as shown in Table 4.4. In this experiment the DENNuRL algorithm showed the ability to generate the best NN to be used as the classification model for any problem, where the only parameter which needs to be modified is the termination condition for that problem.

- Finally, an online phishing email detection system (PEDS) has been proposed to handle zero-day phishing attacks in the online mode. The proposed algorithm encompasses the pre-processing, FEaR, and DENNuRL algorithms discussed

previously in Chapters 3 and 4. An RL-Agent is used to control the system development in the online mode, where it explores new behaviours that may be available in any newly explored dataset. The proposed algorithm registers high accuracy with a very low false positive rate while showing an enhancement of the overall system in terms of a set of metrics in development in the online mode. The experiment has been repeated fifty times, starting in every experiment from a random case, and an average was taken to show the steady performance of the proposed system. The average accuracy registered is 98.6% and the average false positive 1.81%. A full summary of results for all metric is reported in Table 5.3, an example of system development in terms of accuracy is shown in Figure 5.5.

6.3 Difficulties and Solutions

In this study three principal difficulties have been addressed in the phishing email classification. Firstly, the available phishing email dataset is limited, which is very important for the training of the NN to generate PEDS and to assure generality of the proposed model. The proposed model handles this issue by exploring a new behaviour for the correctly classified email with low MSE, and adding that email to upToDate-Dataset which is always growing to reflect the new phishing attacks. Secondly, the phisher is always modifying his techniques to lure the online customer's using new techniques the existing models are never trained how to handle them. Therefore, the proposed model extracts a large number of features that represent most of information available in the email header and content, then the FEaR algorithm is executed to determine the important list of features that can classify emails in the available dataset with acceptable performance. In this strategy, any new behaviour that a phisher tries

to use, the proposed model will be reflected by enabling or disabling a specific feature in the important list of features. Lastly, the performance of our system in the online mode will be very hard to estimate, due to the number of emails being received, where it needs to handle a large number of emails in a short duration. Therefore the proposed framework in the online mode contains two main parts. The first part is the PEDS, responsible for email classification it also determines the right action for each email. The second part is RL-Agent, responsible for monitoring the PEDS output, exploring the new behaviour and deciding whether to replace the existing PEDS with a more powerful one or not.

6.4 Future Work

Based on the research in this thesis, a number of possible future research directions can be identified. These are as follows:

- The adaptation of the NN which used as the core of the detection model can be implemented using Deep Neural Network (DNN). DNN is machine learning framework that can model complex non-linear relationship by using multiple hidden layers. The main Advantages of using DNN are: Firstly, it has the best performance on problems that outperform other techniques. Secondly, it reduces the need for feature engineering, which is one of the most important tasks in the field of phishing email detection. Lastly, the architecture of DNN can be adapted to new problems relatively easily. However, the DNN has some disadvantages that yield to not using it in this thesis such as. DNN is extremely computationally expensive to train. Where the phishing email detection need to

be online that response within a small amount of time. In addition, to use this technique you need to have a very large dataset which is not available in the field of phishing email detection. So, if we can comprehend these two disadvantages the DNN will be a very promising solution in future work.

- After developing the online phishing email detection system, the proposed framework could be used to enhance existing anti-phishing system and it could be employed in email servers to detect phishing attacks.

- The dataset used in developing the PEDS has an important role in building the detection model, and therefore any newly available dataset should be used to enhance the detection model automatically by adding it to the offline dataset.

- The same detection model could be extended from binary classification (phishing and ham email) to a ternary classification model that can classify to spam, phishing, and ham email. This enhancement could be performed by adding more features that represent spam email and the dataset could be updated to include examples of spam emails.

- Finally, the pre-processing algorithm can be enhanced by adding new features to the list of features, which will increase the system's ability to explore any new behaviour using the FEaR algorithm. In conclusion, there are many enhancements that could be taken into consideration in future studies, whether to look more closely at a particular property of phishing email or in extending the classification model for larger domains.

Bibliography

Aamoth, D. (2011). The man who invented email. *Time, November*, 15.

Aarts, B., Chalker, S., and Weiner, E. (2014). *The Oxford dictionary of English grammar*. Oxford University Press.

Abdelhamid, N., Ayesh, A., and Thabtah, F. (2014). Phishing detection based associative classification data mining. *Expert Systems with Applications*, 41(13):5948–5959.

Abu-Nimeh, S. and Nair, S. (2008). Bypassing security toolbars and phishing filters via dns poisoning. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–6. IEEE.

Abu-Nimeh, S. and Nair, S. (2010). Circumventing security toolbars and phishing filters via rogue wireless access points. *Wireless Communications and Mobile Computing*, 10(8):1128–1139.

Abu-Nimeh, S., Nappa, D., Wang, X., and Nair, S. (2007). A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 60–69. ACM.

Aburrous, M. and Khelifi, A. (2013). Phishing detection plug-in toolbar using intelligent fuzzy-classification mining techniques. *Int. J. Soft Comput. Softwa. Eng.(SCSE 2013)*, 3.

Aburrous, M. R., Hossain, A., Dahal, K., and Thabatah, F. (2009). Modelling intelligent phishing detection system for e-banking using fuzzy data mining. In *CyberWorlds, 2009. CW'09. International Conference on*, pages 265–272. IEEE.

Adeoti, O. A. and Osanaiye, P. A. (2013). Effect of training algorithms on the performance of ann for pattern recognition of bivariate process. *International Journal of Computer Applications*, 69(20).

Adler, A. and Schuckers, M. E. (2005). Calculation of a composite det curve. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pages 860–868. Springer.

Aggarwal, C. C. (2007). *Data streams: models and algorithms*, volume 31. Springer Science & Business Media.

Ahamid, I., Abawajy, J., and Kim, T. (2013). Using feature selection and classification scheme for automating phishing email detection. *Studies in Informatics and Control*, 22(1):61–70.

Almomani, A., Gupta, B., Atawneh, S., Meulenberg, A., and Almomani, E. (2013a). A survey of phishing email filtering techniques. *IEEE communications surveys & tutorials*, 15(4):2070–2090.

Almomani, A., Wan, T.-C., Manasrah, A., Altaher, A., Backlizet, M., and Ramadas, S. (2013b). An enhanced online phishing e-mail detection framework based on evolving connectionist system. *International Journal of Innovative Computing, Information and Control (IJICIC)*, 9(3):169–175.

Alsharnouby, M., Alaca, F., and Chiasson, S. (2015). Why phishing still works: user strategies for combating phishing attacks. *International Journal of Human-Computer Studies*, 82:69–82.

Ammar, A. (2014). Dynamic evolving neural fuzzy framework for phishing e-mail detection. *ACM*.

An, N., Zhao, W., Wang, J., Shang, D., and Zhao, E. (2013). Using multi-output feedforward neural network with empirical mode decomposition based signal filtering for electricity demand forecasting. *Energy*, 49:279–288.

Anderson, R., Barton, C., Böhme, R., Clayton, R., Van Eeten, M. J., Levi, M., Moore, T., and Savage, S. (2013). Measuring the cost of cybercrime. In *The economics of information security and privacy*, pages 265–300. Springer.

Ang, J. H., Tan, K., and Al-Mamun, A. (2008). Training neural networks for classification using growth probability-based evolution. *Neurocomputing*, 71(16):3493–3508.

APWG (2015). Phishing activity trends report, 1st âĂŞ 3rd quarters 2015. Report, APWG.

Arachchilage, N. A. G. and Love, S. (2014). Security awareness of computer users: A phishing threat avoidance perspective. *Computers in Human Behavior*, 38:304–312.

Auckenthaler, R., Carey, M., and Lloyd-Thomas, H. (2000). Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1):42–54.

Bache, K. and Lichman, M. (2013). Uci machine learning repository [http://archive. ics. uci. edu/ml]. university of california, school of information and computer science. *Irvine, CA*.

Barraclough, P., Hossain, M. A., Tahir, M., Sexton, G., and Aslam, N. (2013). Intelligent phishing detection and protection scheme for online transactions. *Expert Systems with Applications*, 40(11):4697–4706.

Basu, J. K., Bhattacharyya, D., and Kim, T.-h. (2010). Use of artificial neural network in pattern recognition. *International journal of software engineering and its applications*, 4(2).

Bergholz, A., De Beer, J., Glahn, S., Moens, M.-F., Paaß, G., and Strobel, S. (2010). New filtering approaches for phishing email. *Journal of computer security*, 18(1):7–35.

Berk, R. A. (2016). Classification and regression trees (cart). In *Statistical Learning from a Regression Perspective*, pages 129–186. Springer.

Blocki, J. and Sridhar, A. (2016). Client-cash: Protecting master passwords against offline attacks. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 165–176. ACM.

Boneh, D., Mitchell, J., Ledesma, R., Chou, N., and Teraguchi, Y. (2007). Spoofguard. Technical report, Technical report, http://crypto. stanford. edu/SpoofGuard.

Boulesteix, A.-L., Janitza, S., Kruppa, J., and König, I. R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):493–507.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 38(2):156.

Cantú-Paz, E. and Kamath, C. (2005). An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):915–927.

Caputo, D. D., Pfleeger, S. L., Freeman, J. D., and Johnson, M. E. (2014). Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38.

Castiglione, A., De Prisco, R., and De Santis, A. (2009). Do you trust your phone? In *International Conference on Electronic Commerce and Web Technologies*, pages 50–61. Springer.

Chandrasekaran, M., Narayanan, K., and Upadhyaya, S. (2006). Phishing email detection based on structural properties. In *NYS Cyber Security Conference*, pages 1–7.

Chen, Y., Zahedi, F. M., and Abbasi, A. (2011). Interface design elements for anti-phishing systems. In *International Conference on Design Science Research in Information Systems*, pages 253–265. Springer.

Chou, P.-H., Li, P.-H., Chen, K.-K., and Wu, M.-J. (2010). Integrating web mining and neural network for personalized e-commerce automatic service. *Expert Systems with Applications*, 37(4):2898–2910.

Collier, M. and Endler, D. (2013). *Hacking Exposed Unified Communications & VoIP Security Secrets & Solutions*. McGraw-Hill Osborne Media.

Collinson, H. (1995). America online girds against hacker break-ins. *Computers and Security*, 14(6):519.

Commission, F. T. et al. (2006). An e-card for you game.

Commission, F. T. et al. (2015). Consumer sentinel network data book for january–december 2014.

Costa, L. d. F., Oliveira Jr, O. N., Travieso, G., Rodrigues, F. A., Villas Boas, P. R., Antiqueira, L., Viana, M. P., and Correa Rocha, L. E. (2011). Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412.

Dai, Q. (2013). A competitive ensemble pruning approach based on cross-validation technique. *Knowledge-Based Systems*, 37:394–414.

Del Castillo, M. D., Iglesias, A., and Serrano, J. I. (2007). Detecting phishing e-mails by heterogeneous classification. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 296–305. Springer.

Dhamija, R. and Tygar, J. D. (2005). The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 77–88. ACM.

DoleÅ¿al, B. R. (2008). Anti-phishing firefox plug-in.

Drake, C. E., Oliver, J. J., and Koontz, E. J. (2004). Anatomy of a phishing email. In *CEAS*.

EARTHLINK, I. (2016). Earthlink toolbar.

eBay (2016). Using ebay toolbar's account guard.

El Faouzi, N.-E., Leung, H., and Kurian, A. (2011). Data fusion in intelligent transportation systems: Progress and challenges–a survey. *Information Fusion*, 12(1):4–10.

Engelbrecht, A. P. (2001). A new pruning heuristic based on variance analysis of sensitivity information. *IEEE transactions on Neural Networks*, 12(6):1386–1399.

Ferri, C., Hernández-Orallo, J., and Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38.

Fette, I., Sadeh, N., and Tomasic, A. (2007). Learning to detect phishing emails. In *Proceedings of the 16th international conference on World Wide Web*, pages 649–656. ACM.

Gansterer, W. N. and Pölz, D. (2009). E-mail classification for phishing defense. In *European Conference on Information Retrieval*, pages 449–460. Springer.

GeoTrust, I. (2016). Geotrust trustwatcher toolbar.

Gudise, V. G. and Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 110–117. IEEE.

Gupta, R. and Shukla, P. K. (2015). Performance analysis of anti-phishing tools and study of classification data mining algorithms for a novel anti-phishing system. *International Journal of Computer Network and Information Security (IJCNIS)*, 7(12):70.

Hamid, I. R. A. and Abawajy, J. (2011). Hybrid feature selection for phishing email detection. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 266–275. Springer.

Han, H. and Qiao, J. (2010). A self-organizing fuzzy neural network based on a growing-and-pruning algorithm. *IEEE Transactions on Fuzzy Systems*, 18(6):1129–1143.

Han, M., Guo, W., and Mu, Y. (2007). A modified rbf neural network in pattern recognition. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 2527–2532. IEEE.

Han, W., Cao, Y., Bertino, E., and Yong, J. (2012). Using automated individual white-list to protect web digital identities. *Expert Systems with Applications*, 39(15):11861–11869.

Herzberg, A. and Margulies, R. (2012). Training johnny to authenticate (safely). *IEEE Security & Privacy*, 10(1):37–45.

Hirose, Y., Yamashita, K., and Hijiya, S. (1991). Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4(1):61–66.

Hsieh, C.-C., Liou, D.-H., and Lee, D. (2010). A real time hand gesture recognition system using motion history image. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 2, pages V2–394. IEEE.

Hu, W., Gao, J., Wang, Y., Wu, O., and Maybank, S. (2014). Online adaboost-based parameterized methods for dynamic distributed network intrusion detection. *IEEE Transactions on Cybernetics*, 44(1):66–82.

Huang, H., Zhong, S., and Tan, J. (2009). Browser-side countermeasures for deceptive phishing attack. In *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*, volume 1, pages 352–355. IEEE.

Igel, C. and Hüsken, M. (2003). Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50:105–123.

Inomata, A., Rahman, M., Okamoto, T., and Okamoto, E. (2005). A novel mail filtering method against phishing. In *Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on*, pages 221–224. IEEE.

Islam, M. M., Sattar, M. A., Amin, M. F., Yao, X., and Murase, K. (2009). A new adaptive merging and growing algorithm for designing artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3):705–722.

Islam, R. and Abawajy, J. (2013). A multi-tier phishing detection and filtering approach. *Journal of Network and Computer Applications*, 36(1):324–335.

Islaml, M., Shahjahan, M., and Murase, K. (2000). An algorithm for automatic design of two hidden layered artificial neural networks. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 467–472. IEEE.

Izhar, M., Shahid, M., and Singh, V. (2013). Network security issues in context of rsna and firewall. *International Journal of Computer Applications*, 82(16).

Jakobsson, M. and Myers, S. (2006). *Phishing and countermeasures: understanding the increasing problem of electronic identity theft*. John Wiley & Sons.

Jayalakshmi, T. and Santhakumaran, A. (2010). A novel classification method for diagnosis of diabetes mellitus using artificial neural networks. In *Data Storage and Data Engineering (DSDE), 2010 International Conference on*, pages 159–163. IEEE.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kang, A., Lee, J. D., Kang, W. M., Barolli, L., and Park, J. H. (2014). Security considerations for smart phone smishing attacks. In *Advances in Computer Science and its Applications*, pages 467–473. Springer.

Kang, J. and Lee, D. (2007). Advanced white list approach for preventing access to phishing sites. In *Convergence Information Technology, 2007. International Conference on*, pages 491–496. IEEE.

Kathirvalavakumar, T., Kavitha, K., and Palaniappan, R. (2015). Efficient harmful email identification using neural network. *British Journal of Mathematics & Computer Science*, 7(1):58.

Keivani, F. S., Jouzbarkand, M., Khodadadi, M., and Sourkouhi, Z. K. (2012). A general view on the e-banking. *International Proceedings of Economics Development & Research*, 43.

Khonji, M., Iraqi, Y., and Jones, A. (2012). Enhancing phishing e-mail classifiers: a lexical url analysis approach. *International Journal for Information Security Research (IJISR)*, 2(1/2).

Khonji, M., Iraqi, Y., and Jones, A. (2013). Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121.

Kim, D. W., Yan, P., and Zhang, J. (2015). Detecting fake anti-virus software distribution webpages. *Computers & Security*, 49:95–106.

Kirlappos, I. and Sasse, M. A. (2012). Security education against phishing: A modest proposal for a major rethink. *IEEE Security & Privacy*, 10(2):24–32.

Kobayashi, T. and Okada, H. (2013). The effects of similarities to previous buyers on trust and intention to buy from e-commerce stores: An experimental study based on the svs model. In *IT Enabled Services*, pages 19–38. Springer.

Krombholz, K., Hobel, H., Huber, M., and Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and applications*, 22:113–122.

Kumaraguru, P., Sheng, S., Acquisti, A., Cranor, L. F., and Hong, J. (2010). Teaching johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT)*, 10(2):7.

Kwok, T.-Y. and Yeung, D.-Y. (1997). Constructive algorithms for structure learning in feedforward neural networks for regression problems. *IEEE Transactions on Neural Networks*, 8(3):630–645.

Lastdrager, E. E. (2014). Achieving a consensual definition of phishing based on a systematic review of the literature. *Crime Science*, 3(1):9.

Li, L., Helenius, M., and Berki, E. (2012). A usability test of whitelist and blacklist-based anti-phishing application. In *Proceeding of the 16th International Academic MindTrek Conference*, pages 195–202. ACM.

Likarish, P., Jung, E., Dunbar, D., Hansen, T. E., and Hourcade, J. P. (2008). B-apt: Bayesian anti-phishing toolbar. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 1745–1749. IEEE.

Ludermir, T. B., Yamazaki, A., and Zanchettin, C. (2006). An optimization methodology for neural network weights and architectures. *IEEE Transactions on Neural Networks*, 17(6):1452–1459.

Luo, X. R., Zhang, W., Burd, S., and Seazzu, A. (2013). Investigating phishing victimization with the heuristic–systematic model: A theoretical framework and an exploration. *Computers & Security*, 38:28–38.

Luukka, P. (2011). Feature selection using fuzzy entropy measures with similarity classifier. *Expert Systems with Applications*, 38(4):4600–4607.

Ma, L., Ofoghi, B., Watters, P., and Brown, S. (2009). Detecting phishing emails using hybrid features. In *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on*, pages 493–497. IEEE.

Maggi, F. (2010). Are the con artists back? a preliminary analysis of modern phone frauds. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 824–831. IEEE.

Mao, J., Li, P., Li, K., Wei, T., and Liang, Z. (2013). Baitalarm: detecting phishing sites using similarity in fundamental visual features. In *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, pages 790–795. IEEE.

Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The det curve in assessment of detection task performance. Technical report, DTIC Document.

Mason, J. (2005). The apache spamassassin public corpus.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Maurer, M.-E. (2014). *Counteracting phishing through HCI: detecting attacks and warning users*. PhD thesis, München, Ludwig-Maximilians-Universität, Diss., 2014.

Mazher, N., Ashraf, I., and Altaf, A. (2013). Which web browser work best for detecting phishing. In *Information & Communication Technologies (ICICT), 2013 5th International Conference on*, pages 1–5. IEEE.

Medvet, E., Kirda, E., and Kruegel, C. (2008). Visual-similarity-based phishing detection. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, page 22. ACM.

Miyamoto, D., Hazeyama, H., and Kadobayashi, Y. (2008). An evaluation of machine learning-based methods for detection of phishing sites. In *International Conference on Neural Information Processing*, pages 539–546. Springer.

Mohammad, R. M., Thabtah, F., and McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25(2):443–458.

Mohammad, R. M., Thabtah, F., and McCluskey, L. (2015). Tutorial and critical analysis of phishing websites methods. *Computer Science Review*, 17:1–24.

Moore, T. and Clayton, R. (2007). Examining the impact of website take-down on phishing. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 1–13. ACM.

Na, S. Y., Kim, H., and Lee, D. H. (2014). Prevention schemes against phishing attacks on internet banking systems. *International Journal of Advances in Soft Computing & Its Applications*, 6(1).

Nazario (2015). Phishing corpus.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems*. Pearson Education.

Nguyen, H. H. and Nguyen, D. T. (2016). Machine learning based phishing web sites detection. In *AETA 2015: Recent Advances in Electrical Engineering and Related Sciences*, pages 123–131. Springer.

Nguyen, L. D., Le, D.-N., and Vinh, L. T. (2014). Detecting phishing web pages based on dom-tree structure and graph matching algorithm. In *Proceedings of the Fifth Symposium on Information and Communication Technology*, pages 280–285. ACM.

Olivo, C. K., Santin, A. O., and Oliveira, L. S. (2013). Obtaining the threat model for e-mail phishing. *Applied soft computing*, 13(12):4841–4848.

Oong, T. H. and Isa, N. A. M. (2011). Adaptive evolutionary artificial neural networks for pattern classification. *IEEE Transactions on Neural Networks*, 22(11):1823–1836.

OpenDNS, L. (2016). Phishtank: An anti-phishing site. *Online: https://www. phishtank. com*.

Pamunuwa, H., Wijesekera, D., and Farkas, C. (2007). An intrusion detection system for detecting phishing attacks. In *Workshop on Secure Data Management*, pages 181–192. Springer.

Pandey, M. and Ravi, V. (2012). Detecting phishing e-mails using text and data mining. In *Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on*, pages 1–6. IEEE.

Parekh, R., Yang, J., and Honavar, V. (2000). Constructive neural-network learning algorithms for pattern classification. *IEEE Transactions on neural networks*, 11(2):436–451.

Parmar, B. (2012). Protecting against spear-phishing. *Computer Fraud & Security*, 2012(1):8–11.

Perez, E. and Rendell, L. A. (2016). Using multidimensional projection to find relations. In *Proc. of the Twelfth International Conference on Machine Learning*, pages 447–455.

Ponemon, L. (2015). The cost of phishing & value of employee training. Report, Ponemon Institute.

Prechelt, L. (2012). Early stopping-but when? In *Neural Networks: Tricks of the Trade*, pages 53–67. Springer.

Prechelt, L. et al. (1994). Proben1: A set of neural network benchmark problems and benchmarking rules.

Puma-Villanueva, W. J., Dos Santos, E. P., and Von Zuben, F. J. (2012). A constructive algorithm to synthesize arbitrarily connected feedforward neural networks. *Neurocomputing*, 75(1):14–32.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rader, M. and Rahman, S. (2015). Exploring historical and emerging phishing techniques and mitigating the associated security risks. *arXiv preprint arXiv:1512.00082*.

Raffetseder, T., Kirda, E., and Kruegel, C. (2007). Building anti-phishing browser plug-ins: An experience report. In *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, page 6. IEEE Computer Society.

Ramanathan, V. and Wechsler, H. (2012). phishgillnetâĂŤphishing detection methodology using probabilistic latent semantic analysis, adaboost, and co-training. *EURASIP Journal on Information Security*, 2012(1):1.

Ramesh, R. and Divya, G. (2015). Dynamic security architecture among e-commerce websites. *International Journal of Advanced Computer Research*, 5(19):184.

Ramzan, Z. (2010). Phishing attacks and countermeasures. In *Handbook of information and communication security*, pages 433–448. Springer.

Resnick, P. (2008). Rfc 5322: Internet message format.

Reust, J. (2006). Case study: Aol instant messenger trace evidence. *digital investigation*, 3(4):238–243.

RSA Center, R. A.-F. C. (2013). 2013 a year in review.

Salem, O., Hossain, A., and Kamala, M. (2010). Awareness program and ai based tool to reduce risk of phishing attacks. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 1418–1423. IEEE.

Schäfer, A. M. (2008). Reinforcement learning with recurrent neural networks.

Seera, M. and Lim, C. P. (2014). A hybrid intelligent system for medical data classification. *Expert Systems with Applications*, 41(5):2239–2249.

Sen, S. and Weiss, G. (1999). Learning in multiagent systems. *Multiagent systems: A modern approach to distributed artificial intelligence*, pages 259–298.

Sheng, S., Wardman, B., Warner, G., Cranor, L. F., Hong, J., and Zhang, C. (2009). An empirical analysis of phishing blacklists. In *Proceedings of Sixth Conference on Email and Anti-Spam (CEAS)*.

Smadi, S., Aslam, N., Zhang, L., Alasem, R., and Hossain, M. (2015). Detection of phishing emails using data mining algorithms. In *Software, Knowledge, Information Management and Applications (SKIMA), 2015 9th International Conference on*, pages 1–8. IEEE.

Sowmya, B. and Rani, B. S. (2011). Colour image segmentation using fuzzy clustering techniques and competitive neural network. *Applied Soft Computing*, 11(3):3170–3178.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. (2006). Pac model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM.

Sun, T., Pei, H., Pan, Y., Zhou, H., and Zhang, C. (2011). Neural network-based sliding mode adaptive control for robot manipulators. *Neurocomputing*, 74(14):2377–2384.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

Tankard, C. (2011). Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19.

Thakur, K., Qiu, M., Gai, K., and Ali, M. L. (2015). An investigation on cyber security threats and security models. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*, pages 307–311. IEEE.

Thiyagarajan, P., Venkatesan, V. P., and Aghila, G. (2010). Anti-phishing technique using automated challenge response method. In *Communication and Computational Intelligence (INCOCCI), 2010 International Conference on*, pages 585–590. IEEE.

Tokic, M. (2010). Adaptive $\boldsymbol{\varepsilon}$-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer.

Toolan, F. and Carthy, J. (2009). Phishing detection using classifier ensembles. In *eCrime Researchers Summit, 2009. eCRIME'09.*, pages 1–9. IEEE.

Toolan, F. and Carthy, J. (2010). Feature selection for spam and phishing detection. In *eCrime Researchers Summit (eCrime), 2010*, pages 1–12. IEEE.

Tsamardinos, I., Rakhshani, A., and Lagani, V. (2014). Performance-estimation properties of cross-validation-based protocols with simultaneous hyper-parameter optimization. In *Hellenic Conference on Artificial Intelligence*, pages 1–14. Springer.

UK Cards, F. F. A. (2015). Fraud the facts 2015.

Verma, A. (2013). Effects of phishing on e-commerce with special reference to india. *Interdisciplinary Perspectives on Business Convergence, Computing, and Legality*, page 186.

Wang, J.-H., Wang, H.-Y., Chen, Y.-L., and Liu, C.-M. (2015). A constructive algorithm for unsupervised learning with incremental neural network. *Journal of applied research and technology*, 13(2):188–196.

Wickens, T. D. (2002). *Elementary signal detection theory*. Oxford University Press, USA.

Woodley, A. E. (2012). Resolving the worldâĂŹs commercial disputes: an integrated model for e-learning and odr. *International Journal of Technology Policy and Law*, 1(2):217–233.

Worthy, D. A., Maddox, W. T., and Markman, A. B. (2007). Regulatory fit effects in a choice task. *Psychonomic Bulletin & Review*, 14(6):1125–1132.

Yang, S.-H. and Chen, Y.-P. (2012). An evolutionary constructive and pruning algorithm for artificial neural networks and its prediction applications. *Neurocomputing*, 86:140–149.

Yang, Y., Ault, T., Pierce, T., and Lattimer, C. W. (2000). Improving text categorization methods for event tracking. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 65–72. ACM.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Yao, X. and Liu, Y. (1997). A new evolutionary system for evolving artificial neural networks. *IEEE transactions on neural networks*, 8(3):694–713.

Yu, H., Xie, T., Paszczynski, S., and Wilamowski, B. M. (2011). Advantages of radial basis function networks for dynamic system design. *IEEE Transactions on Industrial Electronics*, 58(12):5438–5450.

Zhang, J., Du, Z.-h., and Liu, W. (2007a). A behavior-based detection approach to mass-mailing host. In *2007 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2140–2144. IEEE.

Zhang, L. and Zhang, B. (1999). A geometrical representation of mcculloch-pitts neural model and its applications. *IEEE Transactions on Neural Networks*, 10(4):925–929.

Zhang, Y., Hong, J. I., and Cranor, L. F. (2007b). Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM.

Zhang, Y., Xiao, Y., Ghaboosi, K., Zhang, J., and Deng, H. (2012). A survey of cyber crimes. *Security and Communication Networks*, 5(4):422–437.

Zhu, W., Zeng, N., Wang, N., et al. (2010). Sensitivity, specificity, accuracy, associated confidence interval and roc analysis with practical sas implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, pages 1–9.

Appendices

Appendix A Code Sample

```
1  package gnusmail.filters;
2
3  import gnusmail.datasource.Document;
4  import gnusmail.datasource.mailconnection.MailMessage;
5  import gnusmail.datasource.mailconnection.MessageInfo;
6  import java.util.StringTokenizer;
7
8  import javax.mail.Message;
9  import javax.mail.MessagingException;
10
11 public final class CompareMsgSenderDomain extends SingleAttFilter {
12
13   @Override
14   protected String getSingleValue(Document document)
15                 throws MessagingException {
16     String res = "False";
17     if (document instanceof MailMessage) {
18                         Message m = ((MailMessage)
     document).getMessage();
19                         MessageInfo mi = new MessageInfo(m);
20                         String contentType= mi.getContentType();
21                         String messageID , messageIDDomain = "";
22                         String from , fromDomain = "";
23
24
25
26                         from = mi.getFrom();
27                         messageID = mi.getMessageId();
28
29                         StringTokenizer st = new StringTokenizer(from, "@
     ");
30                         while (st.hasMoreTokens()) {
31                             fromDomain = st.nextToken();
32                         }
33
34                         StringTokenizer st2 = new StringTokenizer(
     messageID, "@");
35                         while (st2.hasMoreTokens()) {
36                             messageIDDomain = st2.nextToken();
37                         }
38                         if                (messageIDDomain.contains(
     fromDomain.toLowerCase())) {
39                             res = "True";
40                         }
41                 }
42             return res;
43         }
44 }
```

Listing 1: Implementation of CompareMsgSenderDomain

```
function result = Feature_evaluation(input, target)
t = classregtree (input',target');
featureimp = varimportance(t);
%Transpose in column
featureimp = featureimp';
featurenumber = length (featureimp);
%Shift 1 position to dx
featureimp (:,2) = featureimp (:,1);
featureimp (1:end,1) = 1:featurenumber;
%Sort features from the major to the minor.
featureimp = -sortrows (-featureimp, 2);
%Calculate percentage
maximportance = featureimp (1, 2);
featureimp (:,2 ) = featureimp (:,2) / maximportance*100;
FeatureImportanceList = featureimp;
%  view(t)
[nof, ~] = size(FeatureImportanceList);
important_feature = [];
for i = 1 : nof
   if FeatureImportanceList(i, 2) == 0
       break;
   else
       important_feature = [important_feature,FeatureImportanceList(i, 1)
   ];
   end
end
important_feature = sort (important_feature);
result = {};
result = {input(important_feature,:); important_feature};
end
```

Listing 2: Implementation of FEaR Algorithm

```
[x,xt] = size(cell2mat(net.b(1)));
net.trainParam.showWindow = false;
net.trainParam.showCommandLine = false;
net.divideFcn = 'dividerand';
trainingInd =[];
trainingInd = tr.trainInd;
trainingInput = inputs(:,trainingInd);
trainingTarget = targets(:, trainingInd);
[xn, xsettings] = mapminmax(trainingInput);
[tn, tsettings]  = mapminmax(trainingTarget);      % SAVE tsettings
% hidden layer input
b1  = cell2mat(net.b(1));
IW = cell2mat(net.IW(1,1));
[ I, N ]  = size(xn);
B1  = b1*ones(1,N);
% OUTPUT OF INPUT LAYER FOR THE TRAINING DATASET
hidden_layer = tansig(B1+IW*xn);                % SAVE h
LW = cell2mat(net.LW(2,1));           % SAVE LW
hidden_layer = hidden_layer';
%  compute the significant of hidden neuron
Sig_hidden = std(hidden_layer);
Sig_hidden = Sig_hidden / (epoch_counter ^ (1/3));
[ I, N ]  = size(Sig_hidden);
for i = 1 : N
    if Sig_hidden(1, i) < 0.1
        Sig_hidden (2, i) = 1;
    else
        Sig_hidden (2, i) = 0;
    end
end
% Measure the correlation between hidden neuron based on the output of
    the hidden neuron for the training dataset
corr_coff = corrcoef(hidden_layer);
[ I1, N1 ]  = size(corr_coff);
for i = 1 : N1
    corr_coff(i,i) = 0;
end
%  merg least signinficant hiddin neuron with thier most correlated one
% Determine the neuros that will be merged
M1 = 0;
In = 0;
for i = 1 : N
    if Sig_hidden(2,i) == 1
        [M,index] = max(corr_coff(:, i));
        if M > M1
            M1 = M;
            In = index;
        end
    end
end
%    Explore the merging process
% the merge
% input to hidden layer
if and (In ~= 0 , h > 1)
    inputWaight = IW (i, :) + IW (In, :);
    b1 = b1';
```

```
    bais = (b1(i) + b1(In))/2;
    if i < In
        IW(i, :) = [];
        IW(In - 1, :) = [];
        b1(i) = [];
        b1(In - 1) = [];
    else
        IW(In, :) = [];
        IW(i - 1, :) = [];
        b1(In) = [];
        b1(i - 1) = [];
    end
    IW = [IW; inputWaight];
    b1 = [b1 bais];
    b1 = b1';
    % output from hidden layer
    LW = cell2mat(net.LW(2,1));
    outputWeight = (LW(:, i) + LW(:, In))./2;
    if i < In
        LW(:, i) = [];
        LW(:, In - 1) = [];
    else
        LW(:, In) = [];
        LW(:, i - 1) = [];
    end
    LW = [LW outputWeight];
    %  Create the modified neural network

    % First layer
    [n,m] = size(b1);
    layer1_bais = mat2cell(b1, [n], [m]);
    [n,m] = size(IW);
    layer1_weight =  mat2cell(IW, [n], [m]);

    % Second layer
    layer2_bais = net.b(2);
    [n,m] = size(LW);
    layer2_weight =  mat2cell(LW, [n], [m]);

    %     % Third layer
    [h_merg, ~] = size(cell2mat(net.b(1)));
    h_merg = h_merg - 1;
    adapt_net_merg = patternnet(h_merg);
    adapt_net_merg.trainParam.goal = 1e-3;
    adapt_net_merg.trainFcn = 'trainrp';
    adapt_net_merg.trainParam.max_fail = 6;
    adapt_net_merg.trainParam.epochs = epoch_counter;
    adapt_net_merg.divideFcn = 'dividerand';  % Divide data randomly
    adapt_net_merg.divideParam.trainRatio = 70/100;
    adapt_net_merg.divideParam.valRatio = 15/100;
    adapt_net_merg.divideParam.testRatio = 15/100;
    adapt_net_merg.trainParam.showWindow = false;
    adapt_net_merg.trainParam.showCommandLine = false;
    adapt_net_merg.trainParam.epochs = 1;
    [adapt_net_merg, tr1] = train(adapt_net_merg,inputs,targets);
    adapt_net_merg.trainParam.epochs = epoch_counter;
```

```
112     adapt_net_merg.performFcn = 'mse';
113     adapt_net_merg = configure(adapt_net_merg,inputs,targets);
114     adapt_net_merg.IW(1,1) = layer1_weight;
115     adapt_net_merg.b(1) = layer1_bais;
116     adapt_net_merg.LW(2,1) = layer2_weight;
117     adapt_net_merg.b(2) = layer2_bais;
118
119     validationInd =[];
120     validationInd = tr1.valInd;
121     validationInput = inputs(:,validationInd);
122     validationTarget = targets(:, validationInd);
123     validationOutput = adapt_net_merg(validationInput);
124     % Sequre Error Percentage
125     sum = 0;
126     [r, numberOfExample] = size(validationOutput);
127     for i2 = 1 : 2
128         for i = 1 : numberOfExample
129             sum = sum + (validationTarget(i2,i) - validationOutput(i2,i))
    .^2;
130         end
131     end
132
133     merg_sep = 100 * sum *(max(validationOutput(:)) - min(
    validationOutput(:)))/(2*numberOfExample);
134     SEP = [SEP; merg_sep];
135
136     y = adapt_net_merg(inputs);
137     merg_mse_error = mse(adapt_net_merg,targets,y);
138
139     R(current_state, 2) = 1/merg_mse_error;
140  else
141     R(current_state, 2) = -inf;
142
143 end
144
145 %
    ***************************************************************
146 %
147 %
148 %       Expore Neuron addition process
149 % Neuron addition
150 % Take the avearge weight and bais
151 addWeight_layer1 = cell2mat(net.IW(1,1));
152 addBais_layer1 = cell2mat(net.b(1));
153 addWeight_layer2 = cell2mat(net.LW(2,1));
154 addWeight_layer1 = [addWeight_layer1; mean(addWeight_layer1)];
155 addBais_layer1 = [addBais_layer1; mean(addBais_layer1)];
156 addWeight_layer2 = [addWeight_layer2 mean(addWeight_layer2.') '];
157 addBais_layer2 = net.b(2);
158 [n,m] = size (addWeight_layer1);
159 addWeight_layer1 = mat2cell (addWeight_layer1, [n], [m]);
160 [n,m] = size (addBais_layer1);
161 addBais_layer1 =  mat2cell (addBais_layer1, [n], [m]);
162 [n,m] = size (addWeight_layer2);
163 addWeight_layer2 =  mat2cell (addWeight_layer2, [n], [m]);
164 % ********************************
```

```
[h_add p] = size(cell2mat(net.b(1)));
h_add = h_add + 1;
adapt_net_add = patternnet(h_add);
adapt_net_add.trainParam.goal = 1e-3;
adapt_net_add.trainFcn = 'trainbr';
adapt_net_add.trainParam.max_fail = 6;
adapt_net_add.divideFcn = 'dividerand';  % Divide data randomly
adapt_net_add.divideParam.trainRatio = 70/100;
adapt_net_add.divideParam.valRatio = 15/100;
adapt_net_add.divideParam.testRatio = 15/100;

adapt_net_add.trainParam.showWindow = false;
adapt_net_add.trainParam.showCommandLine = false;
adapt_net_add.performFcn = 'mse';
adapt_net_add.trainParam.epochs = 1;
[adapt_net_add, tr2] = train(adapt_net_add,inputs,targets);
%  Set wait and bais of the new neural network
adapt_net_add.IW(1,1) = addWeight_layer1;
adapt_net_add.b(1) = addBais_layer1;
adapt_net_add.LW(2,1) = addWeight_layer2;
adapt_net_add.b(2) = addBais_layer2;
adapt_net_add.trainParam.epochs = epoch_counter;
validationInd =[];
validationInd = tr2.valInd;
validationInput = inputs(:,validationInd);
validationTarget = targets(:, validationInd);
validationOutput = adapt_net_add(validationInput);
% Sequre Error Percentage
sum = 0;
[~, numberOfExample] = size(validationOutput);
for i2 = 1 : 2
    for i = 1 : numberOfExample
        sum = sum + (validationTarget(i2,i) - validationOutput(i2,i)).^2;
    end
end

add_sep = 100 * sum *(max(validationOutput(:)) - min(validationOutput(:))
    )/(2*numberOfExample);
SEP = [SEP; retrain_sep];
y = adapt_net_add(inputs);
add_mse_error = mse(adapt_net_add,targets,y);
R(current_state, 3 ) = 1/add_mse_error;
%
    ****************************************************************
%         add new neuron and relearn neural network

%               new random NN
%
%
%
    ****************************************************************

rng('shuffle');

h1 = (40-5) .* rand + 5 ;
h1 = fix(h1);
```

```
new_net = patternnet(h1);
new_net.trainParam.goal = 1e-3;
new_net.trainFcn = 'trainbr';
new_net.divideFcn = 'dividerand';
new_net.trainParam.epochs=epoch_counter;
new_net.divideParam.trainRatio = 70/100;
new_net.divideParam.valRatio = 15/100;
new_net.divideParam.testRatio = 15/100;
new_net.trainParam.showWindow = false;
new_net.trainParam.showCommandLine = false;
new_net.trainParam.max_fail = 6;
new_net.performFcn = 'mse';
new_net = configure(new_net,inputs,targets);
new_net = init(new_net);
[new_net, tr3] = train(new_net,inputs,targets);
y = new_net(inputs);
new_mse_error = mse(new_net,targets,y);
validationInd =[];
validationInd = tr3.valInd;
validationInput = inputs(:,validationInd);
validationTarget = targets(:, validationInd);
validationOutput = new_net(validationInput);
% Sequre Error Percentage
sum = 0;
[r, numberOfExample] = size(validationOutput);
for i2 = 1 : 2
    for i = 1 : numberOfExample
        sum = sum + (validationTarget(i2,i) - validationOutput(i2,i)).^2;
    end
end

new_sep = 100 * sum *(max(validationOutput(:)) - min(validationOutput(:))
    )/(2*numberOfExample);
SEP = [SEP; new_sep];
R(current_state, 4 ) = 1/new_mse_error;
```

Listing 3: Bart of the implementation of RL-agent that explore a list of actions

Appendix B Publications

Portions of the work within this thesis have been documented in the following publications:

JOURNAL

- **Smadi, S.**, Aslam, N., & Zhang, L. *Detection of Online Phishing Email using Dynamic Evolving Neural Network Based on Reinforcement Learning*, submitted to Decision Support Systems.

CONFERENCE

- **Smadi, S.**, Aslam, N., Zhang, L., Alasem, R., & Hossain, M. A. (2015, 15-17 Dec. 2015). *Detection of phishing emails using data mining algorithms* Paper presented at the 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA).