

Phishing Email and URL Detection using Machine learning and Deep learning

Thesis

Submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

SOMESHA M



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575 025

OCTOBER, 2023

DECLARATION

by the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **Phishing Email and URL Detection using Machine learning and Deep learning** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy in Computer Science and Engineering Department** is a bonafide report of the research work carried out by me. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.


SOMESHA M

Register No. 187105-187CO004

Department of Computer Science and Engineering

Place: NITK, Surathkal.

Date: October 19, 2023


CERTIFICATE

This is to certify that the Research Thesis entitled **Phishing Email and URL Detection using Machine learning and Deep learning** submitted by **SOMESHA M**, (Register Number: 187105-187CO004) as the record of the research work carried out by him, is accepted as the Research Thesis submission in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.


20/10/2023
Prof. Alwyn Roshan Pais

Research Guide

Dr. Alwyn Roshan Pais
(Signature and Seal)
Department of Computer Science & Engineering
National Institute of Technology Karnataka, Surathkal
Mangalore - 575025, Karnataka, INDIA


20/10/23
Chairman - DRPC

(Signature with Date and Seal)

Chairman
DUGC / BPGG / DRPC
Dept. of Computer Science & Engineering
NITK- Surathkal
Srinivasnagar - 575 025

ACKNOWLEDGEMENTS

I am writing to express my sincere gratitude to all the people who supported and inspired me during the tenure of my Ph.D. study. First and foremost, I would like to express my sincere gratitude to my Supervisor Prof. Alwyn Roshan Pais, Professor in the Department of Computer Science & Engineering, for his continuous encouragement, patience, motivation, enthusiasm, and immense knowledge. His guidance and insightful comments helped me in all the research and writing this thesis. Beyond the research work, I have learned a lot from him, such as discipline, management, interpersonal skills, and so on. It will be helpful in the later stages of my career and life. I could not have imagined having a better advisor and mentor for this thesis.

I want to express my sincere thanks and gratitude to my research progress assessment committee members, Dr. Jeny Rajan and Dr. Nagendrappa H., for their positive criticism, insightful feedback, and constructive suggestions throughout my research work. I also extend my sincere thanks to the entire faculty of the Department of Computer Science and Engineering for their support. Nevertheless, I am also grateful to the technical and administrative staff of the department for their timely help and cooperation. Further, I sincerely thank NITK for providing the necessary infrastructure and facilities to complete this research work successfully.

I warmly thank my lab mates Srinivas, Alok, Apoorva, Nikhil, Ajnas, Sivakumar, Zubair, and Vikram for their encouragement and our good times in the department. Further, I thank my fellow batch mates, seniors, and juniors for their support.

I bow to my mother Late. Shankamma M and father Hanuma Naik M for blessing me to achieve my all-set goals. I thank my wife Shashikala and children Jayanth and Diganth, for their incredible support throughout my tenure in all aspects. My special thanks to my brothers Late. Kotresh and Venkatesh, sisters Uma and Vimala, brothers-

in-law, sisters-in-law, Father-in-law, and Mother-in-law, for their continued support and encouragement. My special Thanks to Dr. Nagendrappa & family and Prof. Subhaschandra Kattimani & family, who made my stay at NITK memorable.

I thank all my colleagues at GECK and SKSJTI for their continued support and encouragement.

Special thanks to The Department of Higher Education, Government of Karnataka, and The Director of Technical Education (DTE) Karnataka for permitting me and deputing me to pursue a Ph.D. under the Quality Improvement Program (QIP) scheme.

Finally, thank all of them whose names are not mentioned here but have helped me in some way to accomplish the work.

Somesha M

ABSTRACT

The research thesis attempts to address the issue of email phishing, which poses a serious risk to businesses and corporations. Through the use of social engineering strategies, email phishing assaults persuade users to divulge personal data that can be exploited to access their digital assets. Despite the presence several defenses, the Anti-Phishing Working Group survey reveals that the present approaches to phishing attack detection are still insufficient and ineffective. This underlines the requirement for a more effective system to identify phishing emails and offer greater protection against such assaults to the end user.

There exist many machine learning based techniques to detect phishing emails. Also, they use a large number of heuristics to classify the email. To overcome the disadvantages of existing schemes, we have presented an efficient word embedding cum machine learning framework to classify the emails. The presented technique uses only four email header based heuristics (i.e. From, Return-path, Subject, and Message-ID). The model achieved a significant accuracy of 99.50% using FastText-CBOW algorithm in combination with the Random Forest classifier.

Although machine learning based techniques achieved significant accuracy, it is advisable to use deep learning models whenever we have sufficient data. We have presented an efficient deep learning model called "DeepEPhishNet" for the classification of emails. The presented model based on FastText-SkipGram with Deep Neural Network (DNN) achieved a significant accuracy of 99.52%, TPR of 99.38%, TNR of 99.92%, F-Score of 99.68%, Precision of 99.97%, and MCC of 98.71%.

The above methods make use of only four email header based heuristics for the classification. To study the contribution of the email body in the detection of phishing emails, we have presented an efficient model using transformers. The presented model achieved an accuracy of 99.51% using open source datasets.

The body of the email might contain phishing URLs, which may lead to a phishing attack. In order to overcome this, we have presented an efficient deep learning based

model for phishing URL detection. The accuracy achieved for the DNN, LSTM, and CNN are 99.52%, 99.57%, and 99.43% respectively.

Overall, this research thesis presents efficient techniques for detecting phishing emails and URLs using word embedding, deep learning, and machine learning classifiers.

CONTENTS

List of Figures	xi
List of Tables	xiv
List of Abbreviations	xv
1 Introduction	1
1.1 Information security	2
1.2 Social network	4
1.3 Cyberattack	4
1.3.1 Cyberattack Types	6
1.3.2 Malware	6
1.3.3 Phishing	7
1.3.4 Man-in-the-middle	7
1.3.5 Denial-of-service attack	9
1.3.6 SQL injection	9
1.3.7 Zero-day exploit	11
1.3.8 DNS tunneling	12
1.4 Phishing attack	13
1.4.1 Phishing medium	14
1.4.2 Types of phishing Attacks	15
1.5 Technological Methods for Phishing Attacks	23
1.5.1 Cross-Site Scripting	23
1.5.2 Cross-Site Malicious CAPTCHA Attack	25
1.5.3 Social Engineering	25
1.5.4 QRishing	27
1.6 Phishing attack and prevention	28

1.7	Phishing detection techniques	30
1.8	Need for improving existing techniques to detect phishing attacks	31
1.9	Phishing attack detection challenges	32
1.10	Problem description	33
1.11	Problem statement	33
1.12	Objectives	33
1.13	Thesis Contributions	33
1.14	Thesis Organization	35
2	Literature Review	37
2.1	A Review on Anti-Phishing Types and Techniques	37
2.1.1	ML based phishing email detection	38
2.1.2	Deep learning based email phishing	43
2.1.3	URL Phishing detection using ML	48
2.1.4	URL Phishing detection using Deep Learning	59
2.2	Evaluation Metrics	65
2.3	Datasets used	67
2.3.1	Dataset preparation	67
2.4	Summary	69
3	Phishing email detection framework using word embedding and machine learning	71
3.1	Introduction	71
3.2	Word Embedding:	74
3.2.1	Need of word embeddings:	74
3.2.2	Frequency-based word embedding	75
3.2.3	Prediction-based word embedding	77
3.3	Machine learning	79
3.4	Email phishing	79
3.5	Phishing Email Detection	83
3.5.1	Input Emails	84
3.5.2	Feature Extraction	84
3.5.3	Selected heuristic features	84

3.5.4	Dictionary creation	86
3.5.5	Vectorization	87
3.5.6	Classification	88
3.6	Implementation	89
3.7	Results and discussion	89
3.7.1	Experiment-1	90
3.7.2	Experiment-2	93
3.7.3	Experiment-3	93
3.7.4	Performance Evaluation with Dataset-1	94
3.7.5	Performance Evaluation with Dataset-2	96
3.7.6	Performance Evaluation with Dataset-3	98
3.7.7	Model Validation	98
3.7.8	Performance of individual features	98
3.7.9	Result Analysis	100
3.7.10	Comparison study	100
3.8	Summary	103
4	DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms	105
4.1	Introduction	105
4.2	DeepEPhishNet Framework	109
4.2.1	Classification	110
4.3	Experimental Evaluation	117
4.4	Results and discussion	118
4.4.1	Basic experimental setup	119
4.4.2	Experiment-1: Evaluation of Word2Vec-SkipGram model	120
4.4.3	Experiment-2: Evaluation of Word2Vec-CBOW model	121
4.4.4	Experiment-3: Evaluation of FastText-SkipGram model	122
4.4.5	Experiment-4: Evaluation of FastText-CBOW model	122
4.4.6	Experiment-5: Evaluation of TF-IDF model	123
4.4.7	Models performance analysis with individual datasets	123
4.4.8	Discussion of Bi-LSTM and DNN results	124

4.4.9	Result Analysis	125
4.4.10	Comparison with existing works	127
4.5	Summary	128
5	Phishing Classification based on Text Content of an Email Body using Trans-	
	formers	131
5.1	Introduction	132
5.2	BERT based phishing detection	133
5.2.1	Emails collection	134
5.2.2	In-house dataset preparation	135
5.2.3	Open source dataset collection	135
5.2.4	Data pre-processing	135
5.2.5	Training and classification using transformers	135
5.3	Experimental resources and datasets	137
5.4	Experimental results and discussion	138
5.4.1	Basic experimental setup	138
5.4.2	Results and discussion	138
5.4.3	Result analysis	140
5.4.4	Comparison study	141
5.5	Summary	142
6	Efficient deep learning techniques for the detection of phishing websites	143
6.1	Introduction	144
6.2	Deep learning based URL classification model	149
6.2.1	Feature Extraction	150
6.2.2	Feature Selection	153
6.3	Implementaion	155
6.3.1	Tools Used	156
6.3.2	Datasets Used	156
6.3.3	Deep Learning Algorithms	156
6.4	Results and Discussions	158
6.4.1	Validation of Selected Features using DNN	159
6.4.2	Results with DNN	163

6.4.3	Results with LSTM	166
6.4.4	Results with CNN	167
6.4.5	Result analysis	169
6.4.6	Comparison study	170
6.5	Limitations	172
6.6	Summary	172
7	Conclusion and future work	173
	Bibliography	175
	Publications	194

LIST OF FIGURES

1.1	Internet security	3
1.2	Cyberattack	5
1.3	Malware attack	6
1.4	Life cycle of phishing	7
1.5	Man-in-the-middle attack	8
1.6	DDoS attack	9
1.7	SQL injection attack	10
1.8	Zero-day exploitation	11
1.9	DNS tunneling	12
1.10	Phishing attack	14
1.11	Phishing medium	15
1.12	Spear phishing	16
1.13	Smishing attack	18
1.14	Vishing attack	19
1.15	BEC attack	20
1.16	Email phishing	21
1.17	Cross site scripting	24
1.18	Cross site malicious CAPTCHA attack	25
1.19	Scenario of social engineering attacks	27
2.1	Phishing classification structure	38
2.2	Anti-Phishing techniques based on classification algorithms	39
3.1	Email Phishing attacks from 2010 to 2021	73
3.2	Email message Taxonomy - Courtesy (Almomani et al. 2013)	80

3.3	Email message structure - Courtesy (Almomani et al. 2013)	81
3.4	Architecture of Phishing Email Detection	83
4.1	APWG 2020-21 Phishing Email Statistics	107
4.2	Architecture of "DeepEPhishNet" a Phishing Email Classification Framework	110
4.3	Architecture of LSTM	111
4.4	Bi-LSTM Architecture	113
4.5	Bi-LSTM Experimental Parameters	115
4.6	Architecture of simple neuron	116
4.7	DNN Architecture	118
4.8	DNN Experimental Parameters	119
4.9	Performance of the Model with Dataset-1	124
4.10	Performance of the Model with Dataset-2	125
4.11	Performance of the Model with Dataset-3	126
4.12	Validation Accuracy & Loss graphs - Word2Vec-CBOW cum DNN model with Dataset-1	126
4.13	Validation Accuracy & Loss graphs - Word2Vec-CBOW cum DNN model With Dataset-2	126
4.14	Validation Accuracy & Loss graphs - FastText-SkipGram cum DNN model With Dataset-3	127
5.1	Architecture of the model	134
5.2	BERT - Transformer architecture	136
5.3	BERT base uncased - Example	136
5.4	Accuracy and loss charts for Dataset-I	140
5.5	Accuracy and loss charts for Dataset-II	140
5.6	Accuracy and loss charts for Dataset-III	141
6.1	Architecture of Proposed Model	149
6.2	Network performance using 18 features	162
6.3	Accuracy chart with 14 features	162

6.4	Learning rate with $\alpha= 0.001$ and $\alpha= 0.0001$	162
6.5	DNN Individual feature accuracy	164
6.6	Comparison between Optimizers	165
6.7	DNN accuracy with ten features	165
6.8	LSTM Individual feature accuracy	168
6.9	Accuracy graph of LSTM	168
6.10	Accuracy graph of CNN	169

LIST OF TABLES

2.1	Summary: Email phishing using ML techniques	44
2.2	Summary: Email phishing detection using DL	48
2.3	Summary: URL based phishing detection using ML	57
2.4	Summary: Feature extraction based Phishing URL detection using ML.	58
2.5	Summary: Anti-phishing techniques based on Website phishing using DL	64
2.6	Summary of DL based URL phishing detection using six metrics.	65
2.7	Confusion matrix	67
2.8	Datasets used	69
3.1	APWG Email phishing statistics 2019	72
3.2	APWG Email phishing statistics 2020	74
3.3	Selection of vector size with Dataset-1	91
3.4	Selection of vector size with Dataset-2	92
3.5	Selection of vector size with Dataset-3	95
3.6	Performance Evaluation with Dataset-1	96
3.7	Performance Evaluation with Dataset-2	97
3.8	Confusion Matrix (a). Dataset-1, (b). Dataset-2, (c). Dataset-3	99
3.9	Performance Evaluation with Dataset-3	99
3.10	Summary of the works executed on all three datasets	100
3.11	Performance of individual features	101
3.12	Summary of the works implemented on publicly available datasets.	101
3.13	Summary of the works that used word embedding techniques	103
4.1	Common parameters used in Word2Vec and FastText models	114
4.2	Hyper parameters used in Bi-LSTM	114

4.3	Hyper parameters used in DNN	118
4.4	Performance of Word2Vec-SkipGram model	121
4.5	Performance of Word2Vec-CBOW model	121
4.6	Performance of FastText-SkipGram model	122
4.7	Performance of FastText-CBOW_model	123
4.8	Performance of TF-IDF model	124
4.9	Confusion Matrix (a). Dataset-1, (b). Dataset-2, (c). Dataset-3	125
4.10	Summary of Bi-LSTM and DNN results with word embeddings	127
4.11	Existing works comparison	128
5.1	Used datasets	138
5.2	Model performance with all three datasets	139
5.3	Obtained results using transformers with all three datasets	140
5.4	Comparison Study	142
6.1	Summary of related work in comparison with proposed work	149
6.2	Information gain of individual features	152
6.3	Selected Features	154
6.4	Accuracy of individual features	161
6.5	Parameters for DNN	164
6.6	DNN Experimental Results	164
6.7	Parameters for LSTM	167
6.8	Parameters for CNN	169
6.9	Summary of the results of related existing works	170
6.10	Summary of the works implemented on the same dataset.	171

LIST OF ABBREVIATIONS

<u>Abbreviations</u>	<u>Expansion</u>
AI	Artificial Intelligence
AOL	America Online
APT	Advanced Persistent Threat
APWG	Anti-Phishing Working Group
BEC	Business Email Compromise
BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	Bidirectional LSTM
BNN	Bayesian Neural Networks
CBOW	Continuous Bag of Words
CIA	Confidentiality, Integrity, and Availability
CNN	Convolution Neural Networks
CTSS	Compatible Time Sharing System
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DL	Deep Learning
DKIM	Domain Keys Identified Mail
DNN	Deep Neural Networks
DNS	Domain Name System
DT	Decision Tree
ELM	Extreme Learning Machine
FAIR	Facebook Artificial Intelligence Research
FQDN	Fully Qualified Domain Name
FRS	Fuzzy Rough Set
GCN	Graph Convolutional Network
GRU	Gated Recurrent Unit
HEFS	Hybrid Ensemble Feature Selection
IG	Information Gain
IM	Instant Message
IP	Internet Protocol
IRC	Internet Relay Chat
ISP	Internet Service Provider
ISMS	Information Security Management System

<u>Abbreviations</u>	<u>Expansion</u>
IT	Information Technology
KNN	K-nearest neighbors
LR	Logistic Regression
LSTM	Long Short-Term Memory
MITM	Man-in-the-middle
ML	Machine Learning
MMS	Multi-Message Service
MTA	Mail Transfer Agent
MUA	Mail User Agent
NLP	Natural Language Processing
OCR	Optical Character Recording
PC	Personal Computer
QR	Quick Response
RCNN	Recurrent Convolution Neural Networks
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Networks
SG	SkipGram
SPF	Sender Policy Framework
SQL	Structured Query Language
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
URL	Uniform Resource Locator
VoIP	Voice-over IP
WE	Word Embedding
Wi-Fi	Wireless Fidelity
XSS	Cross-site Scripting

CHAPTER 1

INTRODUCTION

The Internet has become an essential part of everyone's life, serving as a widely used and invaluable resource. It connects countless servers, websites, and web pages. Through it, people can communicate with each other, exchanging emails, images, videos, and messages. In essence, the Internet is a global network of computers and electronics that facilitates the sharing and retrieval of information. Moreover, with a smartphone connected to the internet, users can access a variety of websites, applications, and social media platforms. Arguably, the internet is currently the quickest way to transmit and receive data.

Attachments can be included in emails, such as Microsoft Word documents, PDF files, and scanned copies of paper documents. In theory, there is no limit to the size or quantity of attachments, but in reality, email clients, servers, and Internet Service Providers (ISPs) usually have a maximum limit of 25MB. This discrepancy between what is technically possible and what is allowed can make it hard to know whether a file can be sent via email. For larger files, file hosting services are usually used. Spam constituted up to 30% of all email traffic by 2003 due to the low cost of sending such emails, endangering email's viability as a valuable tool. The US CAN-SPAM Act and similar laws have helped to reduce this issue, though it is still very prevalent. In September 2017, 59.56% of emails were classified as spam. It is expected to rise more than 85% by 2023-24. According to Storm et al. (2017), malicious emails take many forms, such as phishing, email bombardment, and email worms. The first known phishing at-

tempt took place in the mid-1990s when AOL users were tricked into revealing their passwords. To detect phishing attacks, word embedding, machine learning, and deep learning techniques are used (Wollschlaeger et al. 2017). These are more efficient and accurate than existing methods.

1.1 INFORMATION SECURITY

Security of information is a part of information risk management which entails reducing and avoiding unauthorized or improper access to data, misuse, exposure, disruption, deletion, corruption, alteration, recording, or devaluation (Chahid et al. 2017). Additionally, it involves strategies to mitigate the bad effects of such occurrences. The fundamental objective of information security is to maintain the CIA (Confidentiality, integrity, and availability) triad, which refers to ensuring balanced protection of data integrity, confidentiality, and availability, as emphasized by Rybakov and Rybakova (2019). Furthermore, it stresses effective policy implementation without sacrificing organizational efficiency. This is mainly accomplished by using a well-structured risk management procedure that includes the following steps:

- Considering the threats.
- Analyzing the relevant data, resources, and risks, as well as any potential threats, weaknesses, and consequences.
- Selecting a risk management strategy, such as avoiding, mitigating, sharing, or accepting it.
- Keeping an eye on activities and making any required adjustments to deal with any difficulties, changes, or possibilities for improvement.
- Choosing or developing suitable security controls and deploying them in areas where risk mitigation is required.

Academics and practitioners collaborate to create guidelines and regulations to standardize the field, such as rules for passwords, antivirus software, firewalls, encryption software, legal liability, security awareness training, and more. Additionally,

some rules and regulations govern the handling of data, including its access, processing, storage, transfer, and destruction. These rules and regulations, as Miloslavskaya and Tolstoy (2017) suggest, may also play a crucial role in promoting standardization in cybersecurity practices. However, if an organization does not create a culture of continuous development, any standards and guidelines implemented may not have a significant effect. The extensive use of information technology gives rise to possibilities for automating management processes and making services more efficient and of higher quality. Additionally, the use of IT solutions in the public sector emphasizes the need for secure service delivery. As a result, the Information Security Management System (ISMS) is being implemented in public administration institutions to provide security for their information resources and ensure that their mission is fulfilled continuously. The ISMS covers various planning and organizational tasks and is focused on managing information risks that could jeopardize the effectiveness of a public administration institution's operations. Therefore, according to Aldowah et al. (2018), the security management of information in public administration affects the effectiveness, reliability, and quality of public services.

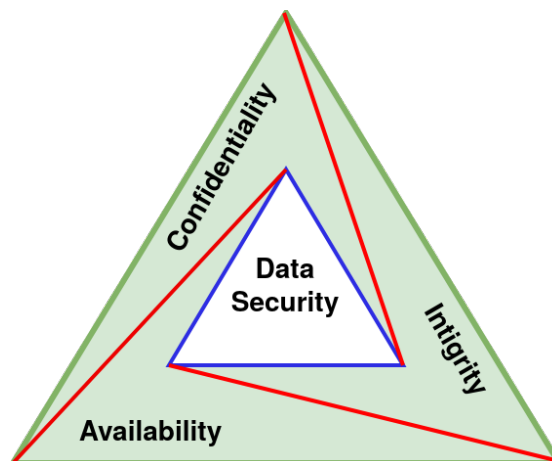


Figure 1.1: Internet security

Figure 1.1 illustrates the Internet security by maintaining confidentiality, availability and integrity to achieve data security in the internet. The goal of information security is to safeguard sensitive data from any unauthorized activity, such as viewing, alteration, recording, obstruction, or eradication. This is done to guarantee the confidentiality and

safety of data like financial data, intellectual property, and client account information.

1.2 SOCIAL NETWORK

Since the start of the twenty-first century, Social media has seen immense growth, allowing people to interact, communicate, and exchange experiences. Twitter, Facebook, and LinkedIn are a few examples that enable users to interact with and discover individuals who share their interests, worldviews, or hobbies. However, the main purpose of these websites is to let users follow the posts made by real individuals. Examples of specialized social media networks that concentrate on particular topics are websites like Pinterest and Tumblr. Since so many people divulge personal information online, phishers can utilize this data to target particular demographics and even get in touch with their victims (Appel et al. 2020).

The social network is a way for social science researchers to understand relationships between people, groups, organizations, and societies. It's a way of thinking about social structures. Every social unit is connected to other members of society through these links. The social network approach suggests that social phenomena should be studied based on relationships between and within units, rather than just the individual properties of units. However, some people criticize this approach because they think it ignores the idea that individuals can act independently and make decisions. Many different types of relations result in different network patterns. This is why network analytics are useful in many different research fields, such as anthropology, biology, communication studies, economics, geography, information science, organizational studies, social psychology, sociology, and sociolinguistics.

1.3 CYBERATTACK

A cyberattack is an assault committed by fraudsters using computational devices on a computer or network of computers and digital devices in the cyber world. Data theft, Disable systems intentionally, or the use of a compromised computer as a launching pad for other attacks are all possibilities in a cyberattack. A cyberattack can be carried out using different methods, such as malware, phishing, ransomware, and denial of service

attacks, as reported by Fang et al. (2019a). Essentially, any action that aims to harm information systems, computer networks, infrastructures, or personal devices can be considered a cyberattack. Those who attempt to gain unauthorized access to restricted data or functionalities in the system with potentially malicious intentions are commonly known as attackers.

Cyberattacks can be categorized as either part of cyberwarfare or cyberterrorism depending on the situation. Figure 1.2 provides a rundown of various cybercrime attacks such as spyware, Domain Name System (DNS) spoofing, and password attacks. Independent individuals, groups, communities, or organizations may launch cyberattacks from an unknown source, using a product referred to as a "cyber weapon" (Eder-Neuhauser et al. 2017). In recent years, the number of cyberattacks has grown significantly, and they can infiltrate a vulnerable system to steal, modify, or destroy the target.

Cyberattacks can take numerous forms, ranging from targeting a single country's infrastructure to infecting a user's PC with malware. Legal professionals limit the term's application to cases where there is actual physical harm to distinguish it from common data breaches and extensive hacking activities. The severity and complexity of cyberattacks continue to increase, making them more dangerous (Sontowski et al. 2020).



Figure 1.2: Cyberattack

1.3.1 Cyberattack Types

There are many cyberattacks identified by researchers are shown in Figure 1.2. Some of them are discussed here,

1.3.2 Malware

Malware, short for malicious software, refers to any type of software program or code designed with the intent of causing harm to a computer system, network, or device. Malware can take many different forms, including viruses, worms, trojans, ransomware, spyware, adware, and more. Malware invades a network by taking advantage of a weakness, most frequently when a user clicks on an email attachment or malicious link, which causes the installation of harmful software (Guillén et al. 2019). Malware has the following capabilities once it has entered the system:

- Installs malicious software or other hazardous programs
- Causes several components to malfunction, rendering the system useless.
- Restricts access to essential network components (ransomware)
- Transmits data from the hard drive to secretly collect information (spyware)

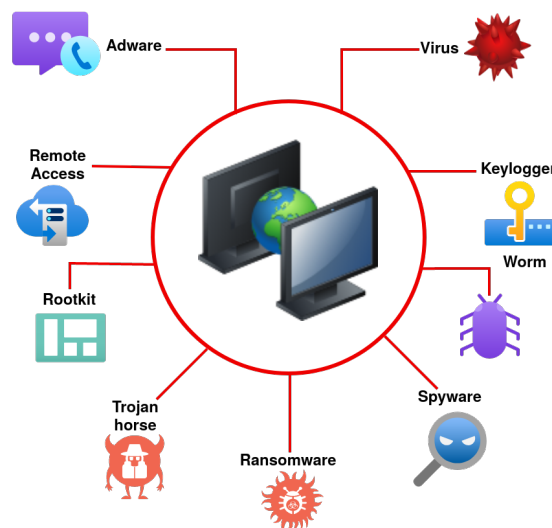


Figure 1.3: Malware attack

Malware can manifest in many forms, such as Trojan horses, worms, ransomware, spyware, rootkit, virus, adware, rogue software, wiper, keylogger, remote access, and

scareware (as illustrated in Figure 1.3). To protect against such malicious software, several defensive strategies should be taken depending on the type of malware. These strategies may include isolating affected systems, installing antivirus programs, activating firewalls, regularly patching systems to prevent zero-day attacks, safeguarding networks from intrusion, and making regular backups of data. Yet as malware becomes more sophisticated, it is designed to elude detection by antivirus software algorithms.

1.3.3 Phishing

A common form of cybercrime is phishing. It was initially found in 1996. Phishing and fishing have a similar tone. This is so because phishing attacks function similarly to fishing in that bait is thrown out for the user to catch. In order to steal the user's personal information, the phisher entices them to visit a phishing website. Figure 1.4 shows the life cycle of phishing. The fraudulent technique of sending emails or other

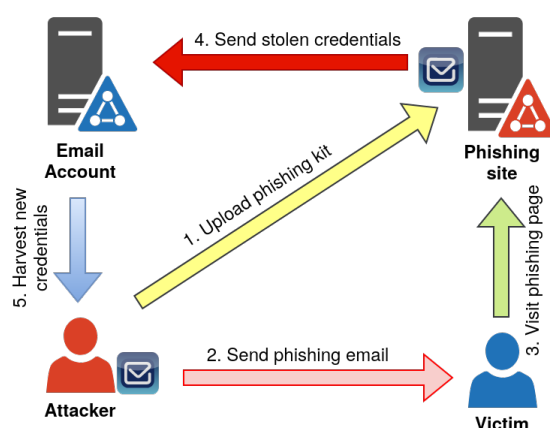


Figure 1.4: Life cycle of phishing

messages that appear to be from a reliable source is known as phishing (Athulya and Praveen 2020). The intention is to spread harmful software on the recipient's computer or acquire access to private data like credit card numbers and login credentials. Phishing has recently increased in popularity as a cyber threat (Kathrine et al. 2019).

1.3.4 Man-in-the-middle

A man-in-the-middle attack is a type of cyber attack where a malicious actor inserts themselves between two parties that are communicating, in order to intercept and po-

tentially modify the communications between them. By intercepting and manipulating the data, the malicious actor can gain access to sensitive information, such as passwords, payment information, or other confidential information (Salem et al. 2021). These are two common scenarios for man-in-the-middle attacks:

1. Once a device has been compromised by malware, an attacker can install software to manipulate all of the victim's data.
2. If a person is on an insecure public Wi-Fi, attackers can position themselves between their device and the network, allowing them to gain access to all the information they have without the visitor being aware.

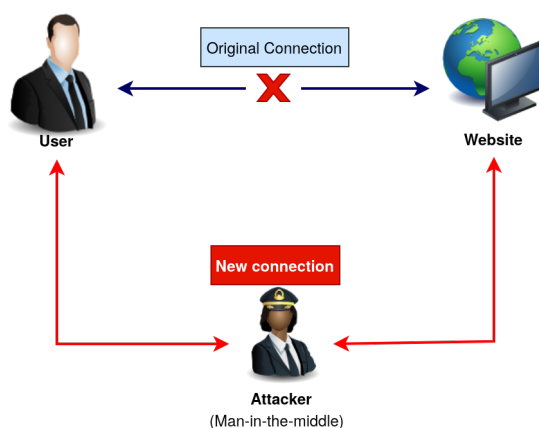


Figure 1.5: Man-in-the-middle attack

As illustrated in Figure 1.5, a man-in-the-middle attack occurs when an intruder inserts themselves into a communication between a user and an application. The purpose of this is to either eavesdrop on the conversation or to pretend to be one of the participants, making it appear that an ordinary exchange of data is happening. The purpose of the attack is to acquire personal information such as credit card numbers, login credentials, and account information. These attacks usually target users of financial applications, cloud-based services, web-based businesses, and websites that require sign-in. The data obtained during an attack can be used for various activities such as fraud, unpermitted trading, or unpermitted password changes. Moreover, it can be used during the infiltration phase of an Advanced Persistent Threat (APT) as a way to build a strong foothold on a secure network.

1.3.5 Denial-of-service attack

A denial-of-service attack is observed when a server, network, or system is flooded with traffic, resulting in the depletion of bandwidth and resources (Jamal et al. 2018). This prevents the system from responding to legitimate requests and can be conducted using multiple compromised devices (Biron et al. 2018). It is also referred to as a Distributed denial-of-service (DDoS) attack.

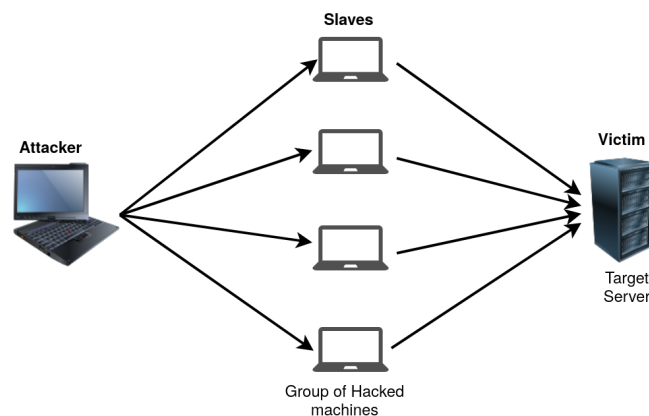


Figure 1.6: DDoS attack

Figure 1.6 illustrates DDoS (Distributed Denial of Service) attack is a type of cyber attack where a website, server, or network is flooded with traffic from multiple sources simultaneously, rendering it unable to function properly or completely shutting it down. In a DDoS attack, a large number of computers or internet-connected devices, known as a botnet, are hijacked by attackers and instructed to send a massive amount of traffic to the target website or server, overwhelming its resources and making it unavailable to legitimate users. The attackers may use a variety of techniques to generate the traffic, such as sending malformed or malicious network packets, initiating large volumes of web requests, or exploiting vulnerabilities in network protocols. DDoS attacks can cause significant disruption to businesses, governments, and organizations, and are a serious threat to the availability and security of online services.

1.3.6 SQL injection

SQL Injection is a type of security vulnerability that occurs in the database layer of an application. It allows an attacker to inject malicious SQL code into a web application's

input fields for execution by the underlying database management system (DBMS). SQL Injection attacks occur when a web application's user-input fields are not properly validated and sanitized. For example, a vulnerable web application might have a login page where users enter their username and password. The application then creates an SQL query to retrieve the corresponding password hash from the database and compare it with the one provided by the user. If an attacker can inject malicious SQL code into the application's input fields, they can modify the SQL query in such a way that it retrieves sensitive information from the database or even allows them to execute arbitrary SQL commands on the underlying database (Hasan et al. 2019). There are many different types of SQL Injection attacks, including simple injection, union-based injection, blind injection, and error-based injection. To prevent SQL Injection attacks, it is important to properly validate and sanitize user input, use parameterized queries or prepared statements instead of constructing SQL queries dynamically, and keep the underlying DBMS and web applications up to date with the latest security patches. Additionally, it is good practice to use least privilege principles, such as using separate database user accounts with limited permissions, to limit the damage that can be done in the event of a successful SQL Injection attack. As depicted in Figure 1.7, attackers can gain control

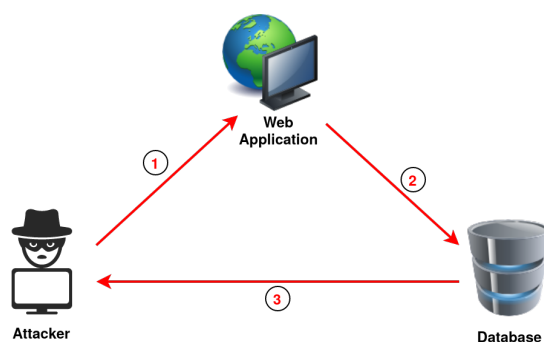


Figure 1.7: SQL injection attack

as administrators of the database server, creating false identities, modifying existing data, leading to issues such as canceling transactions or altering balances, exposing all data stored on the system, destroying data or rendering it inaccessible, and resulting in further repudiation problems.

1.3.7 Zero-day exploit

A zero-day exploit occurs when a network vulnerability is made public but before it can be fixed or patched. This time frame attracts attackers who exploit the newly disclosed weakness. To defend against zero-day threats, ongoing monitoring is essential. The term "zero-day" refers to the fact that a vulnerability or weakness in a software or system is discovered and exploited on the same day that it is first discovered, with "day" referring to the time between when the vulnerability is discovered and when it is publicly known or patched. This means that the software developer has zero days to fix the vulnerability before it can be exploited by attackers. The term is used when the vendor or developer has no time to fix the issue as they just became aware of it. A zero-day attack is when the vulnerability is used by hackers before it can be repaired by engineers. The terms vulnerability, exploit, and attack are closely related to the term zero-day (Kim et al. 2018). A zero-day exploit is a technique used by harmful

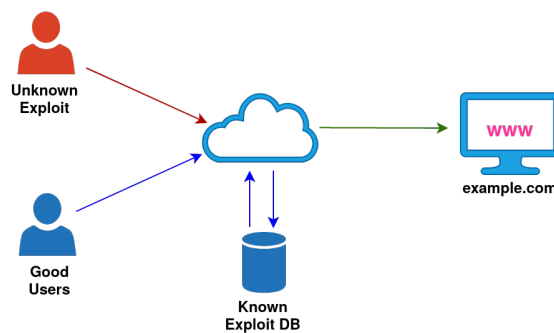


Figure 1.8: Zero-day exploitation

individuals to attack systems that possess a vulnerability. The concept of exploitation is demonstrated in Figure 1.8, where researchers use it to demonstrate how a weakness can be exploited to breach the system or gain unauthorized access. The term "zero-day" refers to the fact that the exploit has not been publicly disclosed or widely known. In some cases, malicious actors may keep the exploit to themselves and use it at a strategic moment. Despite the attacker being aware of the exploit, it remains classified as a zero-day exploit because it is not yet commonly known.

1.3.8 DNS tunneling

DNS Tunneling is a technique that allows the transmission of non-DNS data over the Domain Name System (DNS) protocol. DNS is typically used to resolve domain names into IP addresses and is an essential component of the internet's infrastructure. In DNS tunneling, data is encoded and sent as DNS requests and responses, bypassing firewalls and other network security measures that are not designed to detect such traffic. The technique can be used for both legitimate and malicious purposes. On one hand, it can be used for data exfiltration, bypassing censorship, and bypassing network restrictions. On the other hand, it can also be used to carry out cyber attacks, such as data theft, command and control communication, and malicious data transfers. Malicious actors can manipulate DNS requests to extract sensitive information from compromised systems and transfer it to their infrastructure. Additionally, it can be utilized for communication between the attacker's infrastructure and the compromised system for remote control purposes (Do et al. 2017).

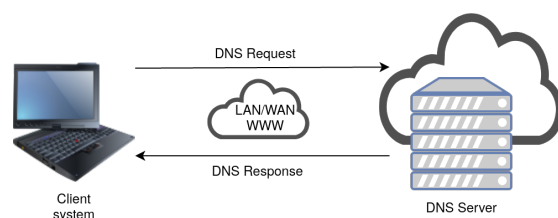


Figure 1.9: DNS tunneling

In Figure 1.9, we can see how DNS tunneling is used to send data and commands between malware and the attacker in the case of targeted attacks that result in data exfiltration. The malware creates a Fully Qualified Domain Name (FQDN), such as "get-command.attacker.com", and sends it as a DNS query to the DNS cache server in the enterprise network. The malware uses this method to request commands from the attacker to search for sensitive information in the enterprise network. The DNS cache server follows the standard procedure to resolve the FQDN by repeatedly querying the attacker.com, root, and com DNS servers. Once the request for a command is received, the attacker.com DNS server sends a response that includes the order to send the malware via the DNS cache server. The malware then repeatedly sends a response

to the command and awaits a new command, ultimately revealing any gathered private information to the outside attacker in a similar fashion.

1.4 PHISHING ATTACK

Phishing is a tactic used by cybercriminals that involve tricking individuals into revealing sensitive information, such as passwords or credit card details. This is done through deceptive means, such as posing as a trustworthy entity in an email, instant message, or text message and enticing the recipient to click on a malicious link. This can lead to the installation of malware on the victim's device, a ransomware attack, or the exposure of confidential information (Guarda et al. 2019). The most common forms of phishing are emails or fake websites. The objective of phishing is to steal personal information for financial gain or to infect a device with malware. To protect oneself, it's crucial to be aware of this type of cyberattack. Phishing attacks are often disguised as being from a reputable source and prompt the victim to enter sensitive information into a fake website (Baykara and Gürel 2018). These attacks can also be used to gather login information for further attacks on a company. Phishing is often the initial step in more complex cyberattacks like APT and ransomware.

Due to the COVID-19 outbreak, people adopted a remote working strategy (work from home). The internet has grown to be a vital tool for establishing social connections, but it also exposes people to deception. Phishing is a type of online fraud where fraudsters try to obtain sensitive data, such as login credentials or credit card information, by impersonating trustworthy entities through emails, text messages, or fake websites. It is not limited to online marketplaces like online markets but can occur through any online communication medium. Phishers use spam emails and false websites that resemble the real ones to entice victims. By fooling users into visiting fake websites through spam emails, they collect personal information.

Phishing attacks are often carried out by creating fake news, such as news about major events, holidays, and anniversaries. The victim receives a message that appears to be from a well-known person or organization. The attack is executed through the use of a malicious file injection or links to malicious websites, with the goal to trick the

victim into revealing personal and financial information, such as passwords, account IDs, and credit card numbers, or leading them to a website where harmful software can be installed on their device. It can be difficult to tell if a message is a successful phishing attempt or not, as these messages often include company logos, other recognizable images, and information obtained from the company. Phishing attacks may also involve the use of subdomains and incorrect Uniform Resource Locators (URLs), similar to other tactics used to manipulate links (Azeez et al. 2021).

Phishing attackers utilize JavaScript to embed a legitimate URL into the browser's address bar. Additionally, JavaScript can be used to alter the URL generated when a link is clicked. User education and training are crucial in identifying phishing communications and serving as the first line of defense against phishing attacks. However, other strategies can also be employed to reduce the likelihood of successful attacks. Figure 1.10 depicts the functioning of phishing, wherein communication messages re-

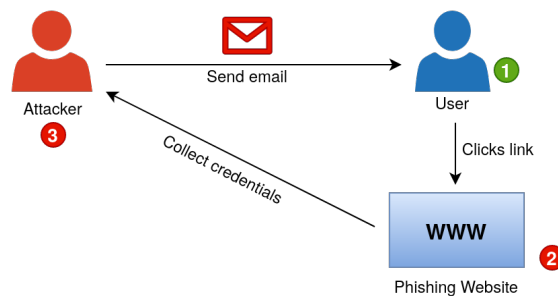


Figure 1.10: Phishing attack

sembling those from trustworthy websites are sent. Phishing emails usually contain links that lead the user to fraudulent websites designed to imitate legitimate sources of information. Subsequently, the user is prompted to provide personal data.

1.4.1 Phishing medium

The initial factor to consider in phishing attempts is the platform or channel used for communication. The medium determines the available technological vectors and strategies. Communication serves as the primary means for phishers to interact with their targets (Kathrine et al. 2019). Phishing mediums refer to the different platforms or channels that fraudsters use to carry out phishing attacks. There are three primary

phishing mediums: the Internet, Short Messaging Service (SMS) / Multi-Messaging Service (MMS), and Voice, as depicted in Figure 1.11. Phishers often use the Internet

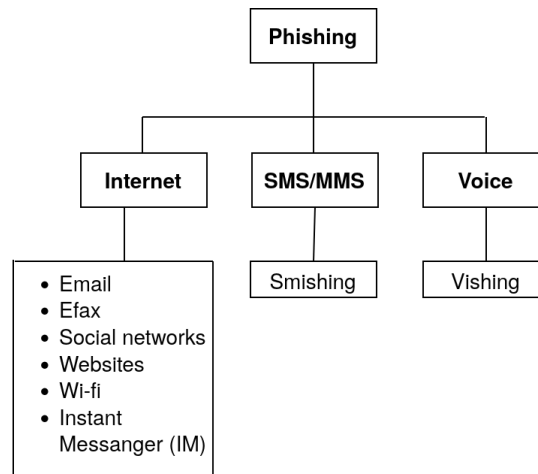


Figure 1.11: Phishing medium

as a medium for phishing due to its vast scope of opportunities. The vector of a phishing attack refers to its starting point, which can be through email, websites, or social media. Technological methods for phishing attacks can be divided into two categories: social engineering and malware-based attacks. Social engineering relies on exploiting a user's fear of losing something valuable to trick them into divulging personal information to the phisher. Meanwhile, malware-based attacks use malicious software to steal information from the victim's device without their knowledge (Al-Hamar et al. 2021).

1.4.2 Types of phishing Attacks

There are many phishing attacks such as spear phishing, Pharming, Whaling and CEO Fraud, Smishing, Vishing, Angler phishing, BEC phishing, Email Phishing, Page hijacking, Efax, Instant Messenger (IM) phishing, and Wi-Fi phishing.

Spear phishing: Spear phishing is a targeted attack that involves crafting and sending emails to a particular individual to make them appear authentic (Yao et al. 2018). Unlike mass phishing, spear phishing attackers utilize information about the intended victim to increase the chances of success. Executives or those in the financial field, who have access to the organization's sensitive data and services, are usually the target of spear phishing (Sonowal 2022b). It is also common for spear phishing to be

used to gain access to an individual's account or to impersonate a high-ranking official or someone involved in key business operations. According to a 2019 study (Burns et al. 2019), accountancy and audit firms are often on the receiving end of spear phishing attacks due to their employees' access to data that could be valuable to criminals. Figure



Figure 1.12: Spear phishing

1.12 illustrates the action of spear phishing. First, the attacker identifies the target and sends the phishing email. The victim opens the email containing malware that is sent by the attacker. When the victim opens the email, the victim's personal data or login details are hacked by the attacker.

Whaling and CEO fraud: When high-level executives, such as the CEO or CFO, are the targets of a whaling attack, valuable data from a corporation is stolen. This could be private personnel information or financial data. High-ranking personnel is a common target of whale attacks due to their influence within organizations and frequent access to confidential information. The term "whaling" refers to the scale of the potential reward for the phishing scam since the "whales" are carefully selected based on their power, access, and influence within the organization (Al-Hamar et al. 2021). Whaling describes spear phishing assaults that are specifically targeted at high-ranking officials and other prominent targets. The content will probably be developed to appeal to the intended audience or role. CEO fraud, which is essentially the opposite of whaling, includes creating forged emails that appear to be from senior executives to persuade other workers at a company to take a particular action, typically moving money to an offshore account. Despite its low success rate, CEO fraud can result in massive financial gains for criminals in the rare instances where it succeeds. Numerous businesses have suffered tens of millions of dollars in losses as a result of such attacks. The attacker's email address may appear to come from a trustworthy source and may even feature business logos or links to a fake website designed to look official. Since

a whale typically holds significant levels of access and trust within their business, it is valuable for the cybercriminal to put in extra effort to make the scam appear credible.

Pharming: Pharming is a type of cyberattack that utilizes malicious software to deceive users into accessing a false website. This can be achieved either by manipulating the host's file on the target's device or by taking advantage of any vulnerabilities in the DNS server program. Computers called DNS servers are in charge of converting Internet names into their corresponding IP addresses. Some people use the term "poisoned" to describe compromised DNS servers. Instead of using a corporate business server, phishing involves unprotected access to a computer, such as changing a customer's home PC. The phrases "farming" and "phishing" were combined to create the term "pharming" (Chavan et al. 2020). An example of a social engineering assault is phishing, which attempts to obtain login credentials such as user names and passwords. Both phishing and pharming have been employed in recent years to gather information for online identity theft. Businesses that host e-commerce and online banking websites are now extremely concerned about pharming. To counter this grave threat, sophisticated anti-pharming methods are necessary. Pharming is not something that antivirus and spyware removal software can stop.

Pharming could happen in one of two ways: either by using a DNS server software vulnerability to the user's advantage or by editing the host file on your victim's PC. Users are purposefully redirected to a false website by cybercriminals in order to obtain and steal usernames and passwords. When a user visits a fraudulent website, malware is downloaded and installed on their computer, damaging information and instigating a pharming attack. When a person uses a browser like Chrome, Firefox, or Opera to visit a website, the browser will contact the DNS server and request the IP address of the domain. If a malicious actor is successful in this phishing attack, the DNS server will be altered.

Smishing: The SMS/MMS medium is to blame for the smishing. Smishing is a type of social engineering attack that uses SMS text messages to trick people into revealing sensitive information or downloading malware. Smishing messages are designed to appear as if they are from a trusted source, such as a bank or a government

agency, and often ask the recipient to click on a link or call a phone number. If the recipient follows the instructions, they may be directed to a fake website that looks legitimate but is designed to steal their personal information, such as login credentials or financial information. Alternatively, the link or phone number may install malware on the recipient's device, allowing the attacker to gain unauthorized access to their device and personal information. Smishing is a serious threat to security and privacy, and it is important to be vigilant when receiving unexpected text messages and to never provide personal information in response to unsolicited requests. An alternative tactic involves sending a target a text message that either has malware embedded in it or contains a link to a malevolent website. Following the installation of the malware, the phisher can continue to carry out their attack, which could range from merely obtaining the target's contacts and messages to creating a botnet or obtaining authorization tokens for logins and transactions (Balim and Gunal 2019; Jia et al. 2021). Figure 1.13 illustrates that

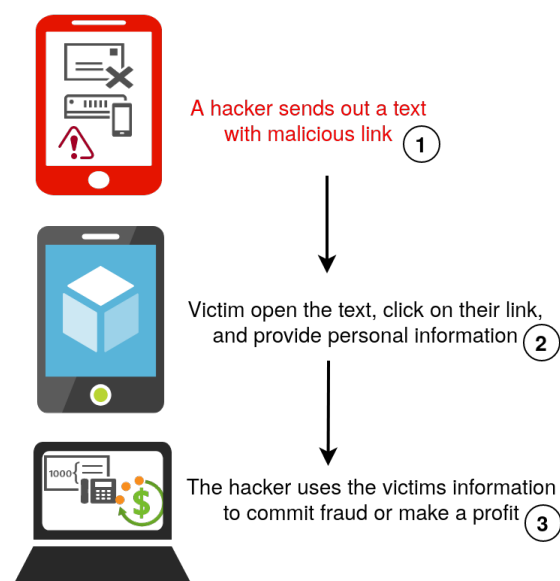


Figure 1.13: Smishing attack

smishing is a three-step process that involves sending a bogus SMS message containing a link. The user will be attacked by the hacker when they click the link and enter their credentials. It is the simplest approach to obtaining someone's data.

Vishing: When victims are tricked over the phone, it is known as a vishing attack, it is also called voice phishing, a new type of crime. That is, the voice-based kind

of phishing is known as vishing. Even while using a phone to try a personal scam is nothing new, the development of voice-over IP (VoIP) technology led to a rise in this activity. Vishing uses number spoofing technology to make calls appear to come from a reliable source. VoIP is used to mask the call's true physical origin, which allows the victim to be tricked into divulging information. This type of deceit has been made easier by VoIP and modern technologies because calls, even international calls, are so inexpensive. Additionally, the usage of automation systems enhances phishers' attacks by blending them into real phone calls (Mondal et al. 2022; Ulfath et al. 2022). Figure

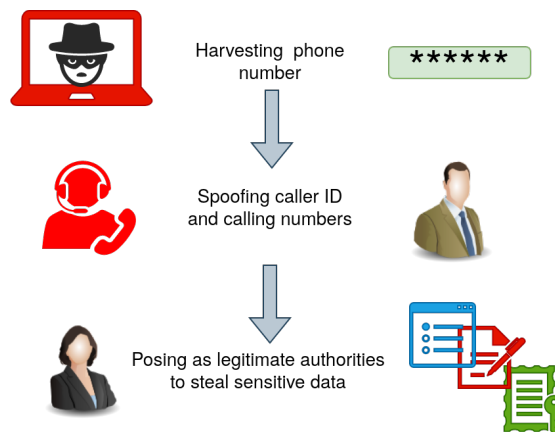


Figure 1.14: Vishing attack

1.14 illustrates a vishing attack flow. The attacker collects the victim's mobile numbers and starts the phishing call to the victim. The attacker first creates the trust of the victim. Then the attacker forces the victim to give the personal data.

Angler Phishing: Using social media sites and accounts, cybercriminals pose as customer service representatives in a new scam called angler phishing. The goal is to deceive disgruntled customers into disclosing personal information. The angler fish, an aquatic animal that pursues other fish, gave rise to the term "angler phishing" assault. Its luminous fin ray attracts prey before it eats them. The same strategies are employed by phishing attackers while fishing for their prey. They construct fictitious social media accounts for prestigious businesses, notably financial institutions. Angler phishers intercept disgruntled customers who try to contact businesses via Twitter, Facebook, or Instagram. To lure victims to fraudulent, attacker-controlled websites, they require them to perform specified tasks (Sonowal 2022a).

Business Email Compromise (BEC) phishing: Cybercrime is a daily threat to businesses and partners of all sizes, and with the quick development of technology and the heavy reliance on it in some transactions, a serious threat type has emerged that poses a high level of risk to businesses and organizations that depend on financial transactions in their operations. BEC, a form of email phishing used for financial gain, is the name of this kind of threat. This attack increased significantly and caused significant financial damages to businesses, particularly during the remote work era and the Corona crisis, as it increased by 94% in the third quarter of this year. This kind of danger simply requires a passable degree of social engineering; it does not demand a significant percentage of knowledge, experience, or abilities in deceit and fraud. BEC is a sort of spear phishing assault that targets only governmental, nonprofit, and commercial organizations having a negative effect (often financial) on those organizations (Al-Musib et al. 2021). As the name suggests, the goal is to access the victim's corporate emails and cause harm using their access; this usually takes the form of data mining and invoice scams. This process, commonly referred to as a launch pad attack, can have a ripple effect in which compromising a single account could lead to the breach or manipulation of a secondary account. Phishers frequently spend weeks or months within a company's networks searching for the ideal attack. This can be accomplished by, for instance, the invoicing system of the company, its suppliers, or a single person (ideally senior management). The attacker then sends a request via email for a money transfer with his or her intentions. The benefit of this form of attack is that the phisher uses another party to arrange the theft rather than stealing money directly. Figure 1.15 illustrates a BEC

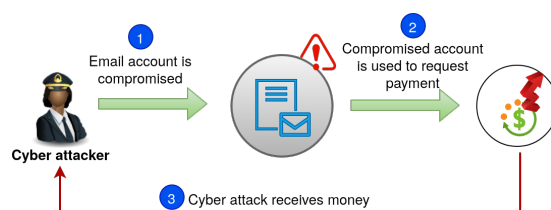


Figure 1.15: BEC attack

attack that targets only governmental, nonprofit, and commercial organizations. The attacker sends the email phishing. The victim opens the email which is compromised. The compromised account is used to request payment. When the victim accepts the

mail and makes the process, the attacker receives the money.

Email Phishing: Email phishing is a type of cyber attack where the attacker tries to trick the recipient into revealing sensitive information, such as passwords, credit card numbers, or other confidential data, by posing as a trustworthy entity in an email message as shown in Figure 1.16. The attacker typically uses social engineering techniques to create a sense of urgency or fear, such as claiming that the recipient’s account has been compromised or that there is an urgent issue that requires the recipient’s immediate attention. Phishing emails are designed to look like they are from a reputable source, such as a bank, a government agency, or a well-known company. They often include logos and other familiar graphics, as well as links that redirect the recipient to a fake website that looks like the real thing. When the recipient enters their information on the fake site, it is collected by the attacker, who can then use it for fraudulent purposes. On darknet markets, compromised streaming service accounts are typically sold directly to customers (Parsons et al. 2019). It is important to be vigilant when receiving emails that ask for sensitive information, and to never enter personal information on a website that is not secure. To protect yourself from phishing attacks, be wary of unsolicited emails, double-check the sender’s email address, hover over links to see where they lead, and be cautious of emails that create a sense of urgency or fear.

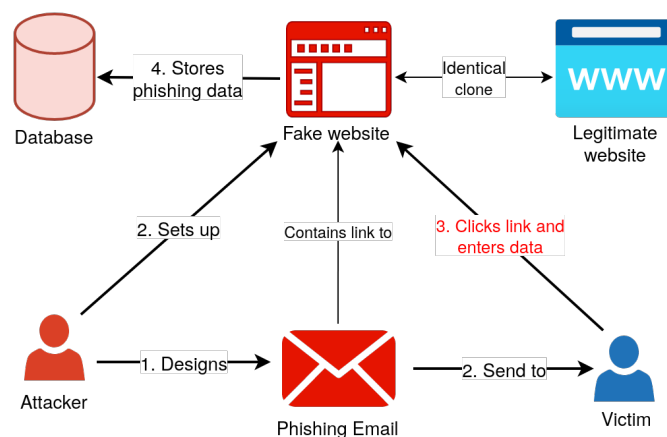


Figure 1.16: Email phishing

Page hijacking: Page hijacking is a type of cyber attack where an attacker gains unauthorized access to a website or web page and replaces the original content with their malicious content. The goal of page hijacking is typically to redirect traffic to a

malicious website, steal sensitive information from users, or spread malware.

There are several methods that attackers use to carry out page hijacking attacks, including:

- **DNS hijacking:** The attacker changes the DNS settings of a website to redirect traffic to a malicious server that hosts the fake web page.
- **Man-in-the-middle (MITM) attack:** The attacker intercepts the communication between the user and the website and injects their content into the communication.
- **Cross-site scripting (XSS):** The attacker injects malicious code into a website or web page that executes in the user's browser when the page is loaded.

To protect against page hijacking attacks, website owners should take several measures, including keeping their software up-to-date, implementing strong authentication mechanisms, and monitoring their websites for unauthorized changes. Users should also be cautious when visiting unfamiliar websites or clicking on links in emails or messages from unknown sources.

EFAX: eFax is similar to a typical fax, however, it doesn't require a fax machine. As opposed to the conventional methods that used phone lines, websites like efax.com use IP (internet protocol) to transfer faxes. The benefit of this approach is that faxes can be delivered to a recipient's device as emails, negating the requirement for a fax machine. However, because this mode of communication takes place online, it creates a new way for phisher attacks to obtain victims' personal information.

Instant Messaging (IM) Phishing: Instant messaging (IM) was one of the earliest methods of online communication, first introduced as IRC (internet relay chat). Other IM systems, including MSN Messenger and Yahoo, were later developed. Today, instant messaging platforms like Facebook Messenger are often combined with other social media platforms, while separate IM platforms like WhatsApp and Telegram, which are not connected to social media, remain widely used. Messages now incorporate emojis, images, gifs, files, and URLs in addition to text, and IM clients can

offer voice and video calling capabilities. Unfortunately, due to its popularity, which surpasses that of SMS messages, IM has become a haven for phishers (Gupta and Singhal 2017). These attackers might use IM phishing tactics, such as sending messages claiming that the recipient has won a prize or suggesting a chat because they feel lonely.

Wi-Fi Phishing: Wi-Fi phishing often takes place in public hotspots, which makes it an untargeted form of a phishing attack. Attackers may choose a specific public Wi-Fi hotspot if they know that a particular target frequently uses it, but this location could also be used as a potential attack vector for spear phishing or whaling. There are various types of Wi-Fi phishing attacks, but the most common one is similar to other phishing methods, where attackers download malicious software to the victim's device to obtain passwords or redirect traffic to fake websites. Additionally, attackers may intercept the data transmitted on these networks, which could lead to the theft of sensitive information from users of the public hotspot (Aravindhnan et al. 2017; Choi et al. 2022).

1.5 TECHNOLOGICAL METHODS FOR PHISHING ATTACKS

The technological methods described below can be used by phishers to access the victim's personal information by exploiting one or more phishing attacks.

1.5.1 Cross-Site Scripting

Modern websites often utilize cross-site scripting to improve user experience, but this also opens them up to cross-site scripting attacks (XSS or CSS). XSS is a type of code injection that is similar to SQL injection, but instead of targeting the query function of databases, it attacks HTML outputs. The code can be written in programming languages such as PHP, NET, or Java. Poorly designed websites that do not properly sanitize user inputs are particularly susceptible to this type of attack, allowing malicious actors to insert their malware. This can result in the code being injected into data fields or the URL. XSS attacks are executed to bypass the same-origin policy (SOP), which restricts scripts loaded from one domain from accessing the data of another domain. As a result, websites should not have access to login credentials or personal information of other domains. However, during an XSS attack, the malicious script is launched as soon as the victim loads the webpage in their browser, allowing the script to access private

information stored in the victim's browser, such as cookies, and send it to the phisher's secure server. This information can then be used by the phisher to enter the user's account and impersonate them (Mohammadi et al. 2017). Figure 1.17 illustrates cross-site scripting. Cross-site scripting (XSS) can occur in two distinct forms: reflected and

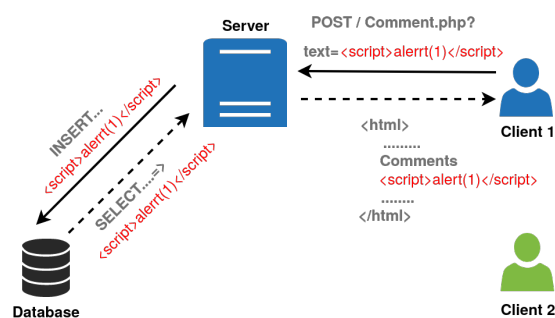


Figure 1.17: Cross site scripting

stored XSS attacks. Of these two, persistent XSS, also known as stored XSS, is the one with a greater impact. This technique involves storing malicious code on a web application server, for example in a database, where it can be accessed by anyone who requests that resource. The attack is not carried out until the victim requests the creation of a dynamic webpage that includes the malicious code. If the code is not sanitized, the victim will load a webpage that has been altered by the attacker. Examples of this type of request include a comment section, blog, or message board. For instance, if a victim accesses a webpage on which the attacker previously posted a comment containing a script that has not been sanitized, every subsequent user's browser that opens the website will run the script, leading to the harvesting and storage of the user's personal information by the attacker until the script is removed. Reflected XSS is the second type of XSS attack. In this case, the script is immediately "reflected" back at the user, rather than being stored indefinitely. The attacker can send the victim a specially crafted link that includes the malicious code as a parameter in an HTTP query. When the victim clicks on the link, the HTTP query is submitted and the malicious code is immediately "reflected" back at the victim in the form of a webpage displaying the results of the query. The victim's private information is taken when the script executes and is sent to the attacker.

1.5.2 Cross-Site Malicious CAPTCHA Attack

By tricking the user into exposing their personal information, the Same Original Policy SOP can also be avoided. In 2016 investigation is done by using a cross-site hostile CAPTCHA assault to achieve this. In this attack, a CAPTCHA is utilized to display user data that has been taken from a trustworthy website. The victim subsequently completes and submits the capture, which gives the phisher access to the victim's confidential information stored on the legitimate website. Alternatives to CAPTCHAs in this type of attack include games and typing tests, or any other format that allows the user's private information to be displayed and communicated to the attacker (Hu et al. 2018). Figure 1.18 illustrates a cross-site malicious CAPTCHA attack. The attacker provides

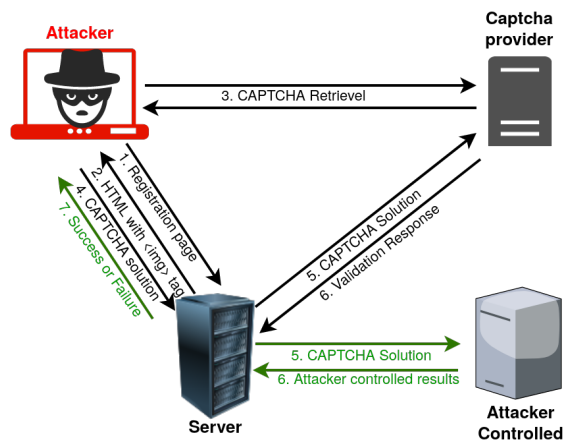


Figure 1.18: Cross site malicious CAPTCHA attack

the fake CAPTCHA to access the victim's personal data. When the victim uses the CAPTCHA their data were hacked by the attacker.

1.5.3 Social Engineering

Social engineering is the art of controlling another person or group of people in order to achieve a goal by taking advantage of their sympathy, generosity, or trust. One of the most established tools in the arsenal of phishers and the larger hacker community is social engineering. The Greek tale of the Trojan Horse is a prime example and what may be called a clever example of social engineering. One of the most adaptable technical techniques is the lack of a predefined media or vector need. It has been described as "the art and science of persuading individuals to do what you want," and it lacks a

specific technical defense tactic. It is possible to classify social engineering capabilities as impersonal employees, hoaxes, confusion-creating tactics, and reverse social engineering. A social engineering attack's main objective is to stop the target from making reasoned decisions and force them to depend on manipulable emotions. This encompasses feelings like "Vanity, Greed, Sense of authority, Anger, Sense of duty, Fear, Sense of belonging, Friendship, Patriotism, Philanthropy".

A phisher can trick a victim into acting irrationally and providing personal information by using these emotions and keeping the target from thinking logically.

Examples of these emotionally-based attacks are the well-known "Nigerian Prince" or "You've won the jackpot" scammers, which prey on potential targets' avarice. These tactics try to influence the target by effectively bribing them, for as by telling a tale about a wealthy person who has money he wants to transfer but needs help. The victim is offered large money in exchange for this assistance, but only after giving the "rich individual" anything, like a small payment or bank account number, physical address, etc., so that a background check may be carried out. The victim is compelled to carry out the phisher's request because of their desire for the substantial sum of money they have been offered. This technique can be used in conjunction with the scarcity-based spear phishing techniques, which claim that if the victim doesn't act, the wealthy person "will find someone else who will," potentially impairing their judgment and leading them to act hastily.

Another sensation that is straightforward to control is a sense of obligation and belonging. If the victim participates in an online group, the phisher, for instance, can pose as a member of the group. The phisher can request that the victim sign a petition or provide money to a cause that the group is based on by suggesting that other group members have already participated. Because signing or contributing will be perceived as a responsibility of all group members, this enhances the possibility that the victim would fall for the scam.

Thirdly, when a phisher poses as an authority figure, people may become afraid. For instance, a phisher may warn a victim that their account would be closed while posing as the organization in charge of that account. An added benefit is if there is a sense of

urgency. Semantic attacks are forms of social engineering that rely on user engagement with computers rather than on direct communication. To compromise a victim's system and steal their personal data, these assaults target the methods by which users interact with their computers. One such attack may involve carefully crafting a phishing website to avoid raising the targets' suspicions. In the majority of phishing attacks across all media and most vectors, the social engineering tactics mentioned above are used (Heartfield and Loukas 2018; Taib et al. 2019). Figure 1.19 illustrates the scenario

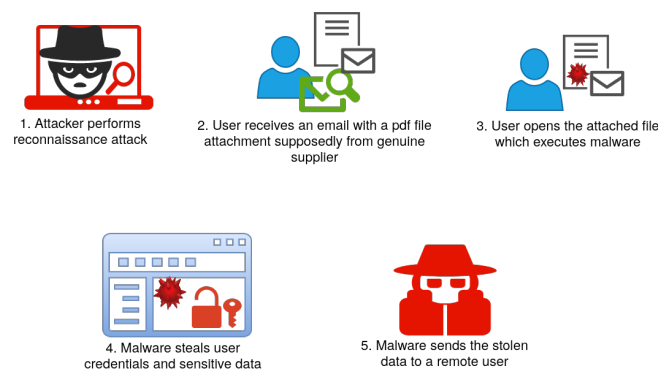


Figure 1.19: Scenario of social engineering attacks

of social engineering attacks. The attacker performs the reconnaissance attack through email. The user receives the email with pdf file attachment, where the a pdf file contains the malware. When the user opens the attached file the malware steals the victim's personal information and sensitive data. Malware sends the hacked data to the attacker.

1.5.4 QRishing

A matrix with a configuration of black and white pixels is used as a QR (quick response) code to store and transmit compressed information. Due to their increased readability and data capacity, two-dimensional QR codes are swiftly replacing outmoded one-dimensional barcodes. An optical scan is used to read QR codes to retrieve the data contained therein, which frequently entails taking a picture of the codes. The information is then processed by a QR code reader after it has been decoded, for instance by opening an app store if the QR code is promoting a new mobile app. Because there are more smart mobile devices on the market, businesses are using QR codes more frequently to point customers to their websites, apps, and items both internally (for

tracking, payment, and discounts) and outside. Nowadays, QR codes are frequently encountered on product packaging, newspaper articles, and billboards.

Unfortunately, the accessibility of creating and disseminating QR codes has made them a perfect tool for phishing assaults. This is made even more effective by the fact that until a QR code scanner decodes it, humans are unable to comprehend its contents. Additionally, many QR code readers launch an URL in a browser or undertake other actions required to view the content of the QR code without first obtaining consent from the user. Keeping with this illustration, a phisher could place QR codes in strategic locations pretending to be advertisements for reputable businesses or products. After that, the malicious URL causes a drive-by download, infecting the victim's device, and the QR codes then reroute scanners of these codes to a trustworthy website. Although the victim wouldn't be aware of the attack, their device would now be compromised and transfer their personal information to the phisher. As an alternative, the link might take visitors to a fake version of the real website, prompt them to log in, and then just grab their login information. The use of URL shortening techniques makes it more difficult for users to tell whether a URL is real, even if the QR code scanner does first give the URL for the victim to check (Dudheria 2017).

In consideration of this, QRishing is an unsafe form of phishing that is simple to combine with the other existing techniques to launch potentially disastrous attacks.

1.6 PHISHING ATTACK AND PREVENTION

Corporate companies commonly implement various security measures, including antivirus software, firewall systems, and email protection, to safeguard their business operations and preserve their identity (Salem et al. 2010). Even while prevention is always the best course of action, a firm should not rely just on its users' best practices. Instead, it should ensure that all of its employees are thoroughly informed about what phishing is and the dangers they run if they are not careful. Email phishing attacks are the most typical type. Phishers occasionally use popular domains like Hotmail or even ones that seem like actual email addresses. If someone you know has fallen victim to a phishing scam, the perpetrators can use his or her account to send spam emails. The

tone used during the attacks is typically scary. Your bank account has been blocked, or Your email account will be erased shortly, are examples of phrases that are used to draw attention and compel the victim to act without thinking.

Here are some tips for preventing phishing attacks:

- User should always verify the email address of the sender.
- User should pay attention to the email's contact details and signature.
- User should not include login information or password on forms or pages that are sent via email.
- Also, users should be cautious when clicking on links and downloading attachments.
- If users receive an unusual request to send money or files from a friend, manager, or coworker, a user should confirm the request with the sender.

When responding to smishing (SMS) messages, victims run the risk of being taken to malicious websites where, after entering his/her information, the data will fall into the hands of criminals. Personal information like address, credit card information and bank login credentials, social media accounts, and emails are typically the targets.

With QRishing, it is possible to attack both people and the systems that will utilize the information contained in this QR Code. It is possible to commit the widest range of frauds depending on the content entered into the QR Code and the security of the application that uses the scanned content. If the QR Code content makes use of a flaw in the application that reads the QR Code, such as a buffer overflow, it may also be able to take control of the victim's device. One benefit for attackers of utilizing a QR code to access a URL is that the user does not have to write the URL and frequently simply retains the presented content, making them vulnerable to phishing attempts that result in websites with identical designs. Users are frequently a system's weakest link, thus training them is essential. When referring to targeted attacks on businesses, users' irrational curiosity exposes them to risk. Users should only use a trustworthy scanner to

scan safe sites, and they should turn off any automatic reader actions, in order to prevent QRishing. There should be an appropriately updated whitelist for information systems, and it should be confirmed that the content size is normal.

Calls made during a Vishing assault may come directly from a person, from recordings, or automated systems. The attacker calls the victim and poses as a representative from a bank or a credit card company. The attacker claims that there has been suspicious activity on the victim's account and asks the victim to verify their identity by providing personal information such as their account number, Social Security number, or date of birth. The attacker may use a spoofed phone number to make it appear as if the call is coming from a legitimate source, adding credibility to their story. If the victim falls for the ruse and provides the requested information, the attacker can use it to gain access to the victim's account or engage in identity theft. The attacker may also use high-pressure tactics, such as threatening to freeze the victim's account or take legal action, to coerce the victim into providing the information.

To protect against vishing attacks, it's important to be cautious when receiving unsolicited phone calls and to never give out sensitive information over the phone unless we are certain of the caller's identity. If we receive a suspicious call, hang up and call the company's customer service number directly to verify the legitimacy of the request.

1.7 PHISHING DETECTION TECHNIQUES

Many phishing detection techniques have been developed by researchers using various approaches, some of which are as follows:

- **DNS Analysis:** Examining DNS records to identify malicious domains and other unusual activity.
- **DNS blocking:** preventing emails from being sent from domains known to be used in phishing attacks.
- **IP Reputation Checks:** This involves comparing the IP addresses of suspicious emails to databases of known malicious IPs.
- **URL Analysis:** Searching the source code of a website or email for suspicious

links, malicious scripts, and other phishing-related indicators.

- **URL Scanning:** Detecting malicious domains and links by scanning the URLs of suspicious emails.
- **Attachment Analysis:** Examining email attachment content to detect malicious files.
- **Keyword Filtering:** Configuring keyword filters to detect phishing emails that contain specific words or phrases.
- **Content Analysis:** Examining email content for potentially phishing-related phrases, words, links, and images.
- **Machine Learning:** Detecting suspicious patterns in emails, websites, and other data using machine learning algorithms.
- **Web Browser Protection:** Installing browser extensions and plugins to detect and warn about malicious websites.
- **Email Authentication:** Checking emails for Sender Policy Framework (SPF), Domain Keys Identified Mail (DKIM), and Domain-based Message Authentication, Reporting, and Conformance (DMARC) compliance.
- **Behavioral Analysis:** Examining user behavior for deviations from normal behavior that could indicate a phishing attack.
- **Anti-Virus Software:** Scanning emails for viruses, malware, and other malicious code.
- **User Education:** Educating users on how to recognize and report phishing emails.

1.8 NEED FOR IMPROVING EXISTING TECHNIQUES TO DETECT PHISHING ATTACKS

The internet has become an essential part of daily life for everyone. Banking, booking, and recharging are now commonplace online transactions. However, these activities

raise the risk of phishing attacks, which are the leading cause of online security problems. Because of the increasing frequency of attacks, phishing has become a major concern for global internet security and the economy. User education is essential for increasing technical awareness and decreasing susceptibility to phishing attacks. While many phishing strategies and attacks exist, current detection techniques are insufficient. We proposed using word embedding, deep learning, and machine learning techniques to classify phishing emails, as well as an effective deep learning technique to detect phishing websites, to improve performance.

1.9 PHISHING ATTACK DETECTION CHALLENGES

Many machine learning algorithms, as well as deep learning systems, cannot process strings or plain text in their raw form. They require numbers as inputs to perform any type of work (classification, regression, etc.). In order to create useful applications, knowledge must be extracted from large amounts of text-based data. Commercial firms' sentiment analysis of reviews, Google's document or news classification or clustering, and so on are examples of real-world text-based applications. Word embedding refers to a collection of language models and feature selection techniques that are commonly referred to as word representation. Its primary goal is to map textual terms or phrases into a continuous, low-dimensional space. Word embeddings effectively convert human-written discourse into numerical form. It's possible that the language that was converted to numbers now has a new numeric representation. Deep learning, a more efficient technique, can be used to detect phishing attacks on websites. A combination of machine learning, deep learning, and word embedding techniques is used to classify email phishing attacks. Researchers can improve these techniques by adding heuristic capabilities to detect phishing emails and websites. Identifying websites hosted on compromised domains, as well as those embedded with flash, HTML, and iframes, may also be part of this process. These additional capabilities would improve the overall effectiveness of detecting phishing attacks.

1.10 PROBLEM DESCRIPTION

Phishing is a type of manipulation attack in which the attacker disguises himself as a trustworthy entity in order to trick victims into disclosing personal information. It is a serious threat because it exploits human rather than system vulnerabilities. Users must be educated and trained to recognize phishing websites and emails, but expecting everyone to do so perfectly is unrealistic. User education and anti-phishing tools are ineffective because attackers are constantly devising new ways to exploit flaws in existing systems. While user education is important, it is not sufficient on its own. As a result, we require a more effective and highly accurate system for detecting phishing emails and websites.

1.11 PROBLEM STATEMENT

Design and develop an efficient mechanism to detect phishing websites and emails with high accuracy by using limited features.

1.12 OBJECTIVES

- Propose an efficient word embedding framework to detect phishing emails using email header features.
- Propose an efficient NLP based deep learning framework to detect phishing emails using email body text.
- Propose an efficient deep learning classification mechanism to detect phishing websites using minimal distinctive URL features.

1.13 THESIS CONTRIBUTIONS

The thesis proposes the use of word embedding, machine learning, and deep learning methods to enhance the effectiveness of detecting phishing attempts in emails and websites. Word embedding represents words in a numerical format for better understanding, while machine learning algorithms analyze data to learn from patterns and trends, and deep learning techniques involve complex neural networks for pattern and feature detection. Using these techniques together can improve phishing detection efficiency and

accuracy to protect individuals and organizations from such attacks. The main contributions of the thesis are as follows:

- Proposed an efficient phishing emails detection technique based on header features using word embedding and machine learning (Somesha and Pais 2022). The proposed system uses a novel word embedding cum machine learning framework to classify emails using only four email header-based heuristics (From, Return-path, Subject, and Message-ID). In order to create precise email anti-phishing systems, a real-time input data set is required. In this study, a real-time in-house collection of phishing and legitimate email datasets are created.
- Proposed "*DeepEPfishNet*" a deep learning framework for email phishing detection using word embedding algorithms. The technique also makes use of four header-based features of the emails for email classification. Various word embeddings have been evaluated, and the model based on FastText-SkipGram with DNN achieved an accuracy of 99.52%.
- Proposed a model which utilizes BERT transformers to analyze the text content of email bodies, enabling accurate identification and classification of phishing attempts. By leveraging the power of BERT transformers, the model effectively captures context, semantics, and syntactic structures, leading to enhanced performance in phishing detection and classification with an accuracy of 99.51%.
- Proposed an efficient deep learning model for the detection of phishing websites (Somesha et al. 2020). The framework relies on minimal distinctive URL features to address the issues associated with phishing websites and to improve accuracy and efficiency in phishing detection. The model incorporates Deep Neural Network (DNN), Long Short-Term Memory (LSTM), and Convolution Neural Network (CNN). Among all three methods, LSTM achieves an accuracy of 99.57%.

1.14 THESIS ORGANIZATION

The thesis is organized as follows: Chapter 2 gives a survey of various techniques currently employed for the detection of phishing emails and websites. Chapter 3 introduces a framework that utilizes word embedding and machine learning, focusing on header features to identify phishing emails. It also covers the procedures for preparing an in-house dataset. Chapter 4 delves into deep learning techniques specifically designed for the detection of phishing emails, utilizing the four header heuristics previously selected and discussed in Chapter 3. Chapter 5 proposes a method for classifying phishing emails based on the textual content of their bodies, employing BERT transformers. Chapter 6 describes the implementation of efficient deep learning techniques for identifying phishing websites. Lastly, Chapter 7 concludes the thesis and explores future research directions in the field of phishing detection.

CHAPTER 2

LITERATURE REVIEW

Numerous studies have been conducted using various methodologies to detect phishing emails and websites. This chapter provides a thorough examination of anti-phishing techniques designed specifically to combat email and website phishing. Furthermore, the chapter includes a plethora of phishing indicators corresponding to each phishing category as well as their associated limitations.

2.1 A REVIEW ON ANTI-PHISHING TYPES AND TECHNIQUES

At present, the primary focus of browsers, antivirus software, and existing anti-phishing techniques is to safeguard online users from phishing attacks. They employ a range of tools and techniques to detect both email phishing and website phishing attempts. These protective measures include the use of whitelists, blacklists, source code analysis, URL validation, examination of images and logos, scrutiny of document object models, evaluation of search engine results, assessment of page ranking, and analysis of WHOIS data. We categorize these anti-phishing techniques based on the classification mechanisms they employ, and in the following discussion, we explore each category along with its limitations. Figure 2.1 illustrates the classification of defense mechanisms against email and website phishing. Researchers utilized classification algorithms or models (shown in Figure 2.2) for implementing classification mechanisms in their study. In this survey, our primary focus is on Machine Learning (ML) and Deep Learning (DL) techniques, which have predominantly excelled in the detection of

phishing attacks.

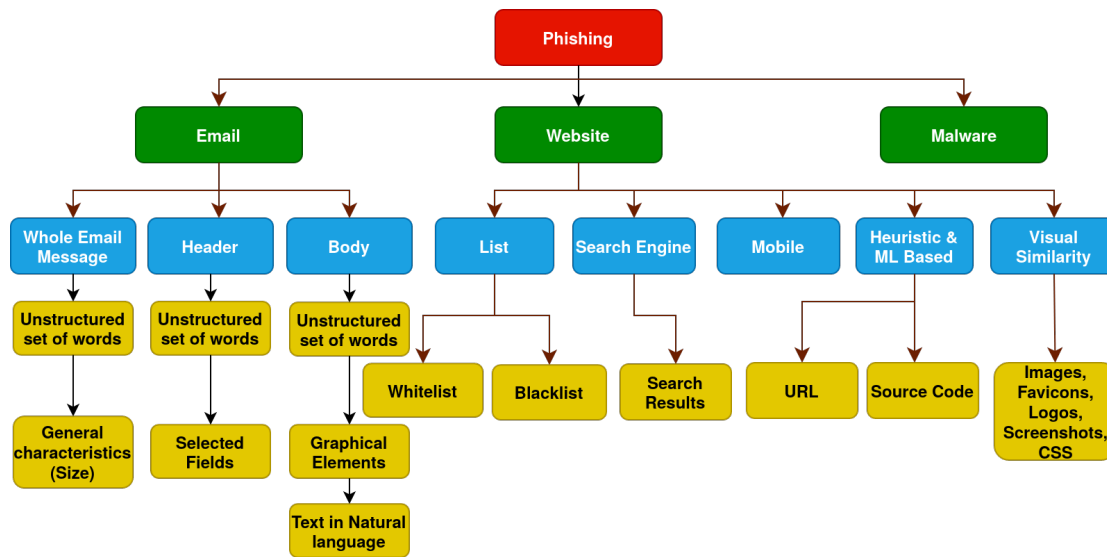


Figure 2.1: Phishing classification structure

2.1.1 ML based phishing email detection

Anti-phishing mechanisms are critical in reducing the risks associated with email phishing attacks. These mechanisms are intended to detect and prevent phishing attempts, keeping users safe from fraudulent emails. Email filtering, link and URL analysis, sender authentication, content analysis, and user education are all common anti-phishing techniques. To create effective anti-phishing tools and techniques, we need technical expertise as well as a thorough understanding of phishing techniques. ML, DL, Word Embedding (WE), Natural Language Processing (NLP), Artificial Intelligence (AI), and phishing website analysis are common techniques used in the development of anti-phishing tools.

Fette et al. (2007) provide a thorough literature review on the detection of phishing emails, which are deceptive messages designed to obtain sensitive information from users. The authors investigate the characteristics of phishing emails as well as the need for effective detection mechanisms to combat this ever-changing threat. They go over various types of features used in phishing email detection, such as content-based, URL-based, header-based, and behavioral-based features. The authors also discuss various machine learning techniques for classification and anomaly detection, such as super-

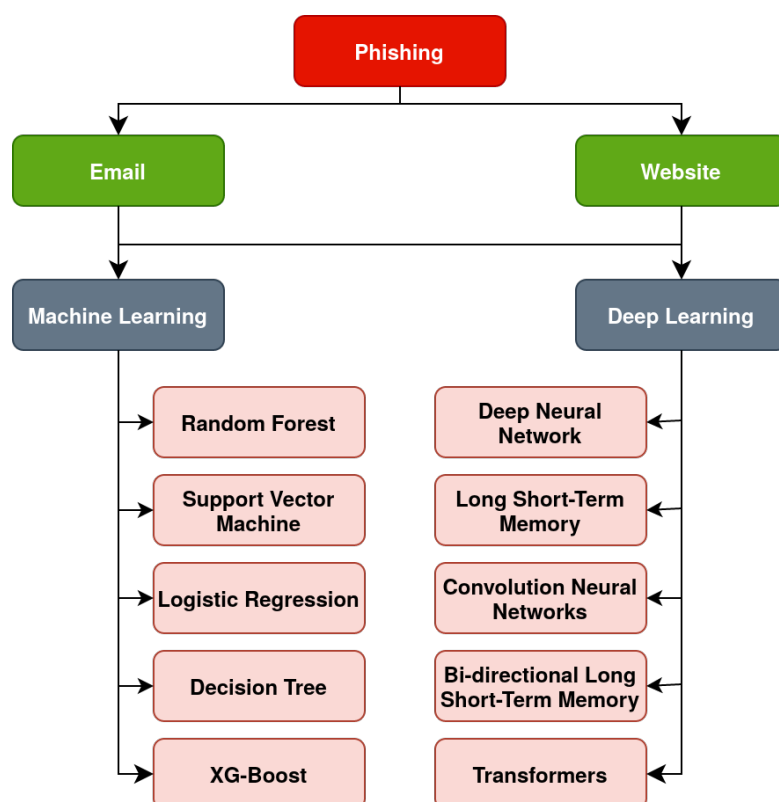


Figure 2.2: Anti-Phishing techniques based on classification algorithms

vised and unsupervised learning. The importance of diverse datasets and evaluation metrics are emphasized. The data is a valuable resource for researchers and practitioners working to develop robust systems for detecting and mitigating phishing emails.

Smadi et al. (2018) proposed a framework for detecting phishing attacks in the on-line mode that combines neural networks and reinforcement learning. The proposed model can dynamically adapt to identify newly arrived phishing emails for newly explored email behaviors. A novel algorithm is used to investigate new phishing behaviors in new datasets. The dynamic system achieves 98.63% accuracy, 99.07% TPR, and 98.19% TNR. The disadvantage of this approach is that learning dynamic updates of features and datasets for each email may cause the system to slow down.

Toolan and Carthy (2009) proposed an extension to the work of Fette et al. (2007), using classifier ensembles for the classification of phishing and non-phishing emails. They used the C5.0 algorithm and achieved a very high precision. Toolan and Carthy (2009) used only FIVE features on approximately 8000 emails, half of which were

phishing and remaining legitimate.

Bergholz et al. (2010) proposed new filtering approaches by selecting novel features suitable to identify phishing emails. The chosen features suites better to statistical models of low dimensional descriptions of email topics. The work was carried out by sequential analysis of email text, external links, and detection of embedded logos as well as indicators for hidden salting (inserting white text on white background). They used 27 basic features with two novel features (logo detection and hidden salting) and obtained an f-measure of 99.46%.

Toolan and Carthy (2010) identified 40 features extracted from the email body of over 10,000 emails which are divided into ham, spam, and phishing. The selected features are evaluated using an information gain algorithm and classified as Best-IG, Median-IG, and Worst-IG features. Best-IG features outperformed among all with an average accuracy of 97.1%. The freely available datasets from SpamAssassin and Phishing corpus was used (4202-ham, 1895-spam, and 4563-phish).

Khonji et al. (2011) proposed feature subset evaluation and feature subset searching methods. The primary focus of this is to enhance the classification accuracy of phishing emails by finding the effective feature subsets from the number of previously proposed features. There are a total of 21 features selected (email body, email header, URL, JavaScript, and external features) from Fette et al. (2007), Bergholz et al. (2010), Toolan and Carthy (2010), and Gansterer and Pölz (2009). After evaluating with various feature selection methods, Wrapper with RF performed the best with 21 features and an f1-score of 99.396%. The authors used publicly available datasets of 4116 phishing emails from monkey.com¹ and 4150 ham emails from SpammAssasin.com².

Abu-Nimeh et al. (2009) proposed distributed phishing detection by applying variable selection using Bayesian Additive Regression Trees (BART). They presented a distributed client-server architecture to detect phishing e-mails by automatic variable selection. BART improves its predictive accuracy when compared to other classifiers. This architecture is also used to detect phishing attacks in a mobile environment. Abu-

¹<https://monkey.org/jose/phishing/>

²<https://spamassassin.apache.org/old/publiccorpus/>

Nimeh et al. (2009) used 71 features for training and testing of 6 Machine learning algorithms (RF, LR, SVM, Nnet, CART, BART), and proved that there is no standard classifier for phishing email prediction.

Chandrasekaran et al. (2006) proposed a technique to classify phishing based on the structural properties of phishing e-mails. They used one-class SVM to classify phishing e-mails before it reaches the user's inbox, essentially reducing human exposure based on selected features. The prototype sits between user's Mail Transfer Agent (MTA) and Mail User Agent (MUA) and process each arriving email. Their results claim a detection rate of 95% of phishing e-mails with a low false positive rate.

Cohen et al. (2018) proposed a novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods. The proposed features are extracted directly from the email itself, therefore the features are independent, do not require the internet or any other tools, and meet the needs of real-time systems. These features are from all components, i.e., header, body, and attachments. The authors used 33142 emails which contain 38.73% of malicious and 61.27% benign emails. Applied 30 most prominent features of the 100 features extracted by applying three main feature selection approach those are, Filter methods, wrapper methods, and embedded methods. Random Forest (RF) classifier achieved the highest detection accuracy of 92.9%, TPR 94.7%, FPT 0.03 among 9 commonly used machine learning classification algorithms (J48, RF, NB, Bayesian Networks, LR, LogitBoost, Sequential Minimal Optimization, Bagging, and Adaboost).

Harikrishnan et al. (2018) made use of TF-IDF and some classical machine learning algorithms such as RF, AdaBoost, NB, DT, and SVM. The proposed method uses TF-IDF for vector representation of words and SVD, NMF for feature extraction, and dimensionality reduction. This model is trained on IWSPA-AP 18 datasets. The proposed model has a testing accuracy of 90.29% for emails with headers using TF-IDF and NMF representation.

Valecha et al. (2021) used a new convention called Persuasion cues instead of features, keywords, or phishing techniques used by other researchers. The proposed tech-

nique uses Word2Vec with four machine learning classifiers and compared the candidate model for gain, loss, and gain_loss persuasion cues with the baseline model and achieved an improvement of approximately 5 to 20%. The model with Word2Vec and SVM achieved the highest gain accuracy of 96.52%, loss of 96.16%, and gain_loss accuracy of 95.97%.

Oña et al. (2019) proposed a novel approach to identify and mitigate phishing attempts. Their method involves utilizing Feature Selection, Automatic Learning, and Neural Networks with Scrum methodology to develop a system that can recognize and respond to phishing attacks that are recorded on an email server. The system was validated using the blacklist of Phish Tank, a collaborative resource for information on internet-based phishing. Their proof of concept demonstrated that the feature selection algorithm effectively eliminates irrelevant email properties, and the neural network algorithm efficiently learns and processes the relevant ones. Future studies could explore the use of deep learning techniques and Bayesian Neural Networks (BNN) to enhance this approach.

The researchers (Moradpoor et al. 2017), utilized publicly available email datasets consisting of benign and phishing emails to construct a neural network (NN) based model aimed at detecting and categorizing phishing emails. The datasets contained genuine emails extracted from the "Spam Assassin" and "Phishcorpus" datasets, respectively. The datasets were designed to include emails of varying difficulty levels. For example, certain innocuous emails present in the "Spam Assassin" dataset could be easily distinguished from phishing emails due to the absence of phishing signatures. In the future, research may be conducted to explore word embedding techniques, which may help enhance the model's performance and improve its accuracy and efficiency in detecting phishing techniques.

Researchers led by Baykara and Gürel (2018) have created an anti-phishing simulator that can recognize phishing attacks in both emails and websites. The simulator offers information on how to detect phishing emails and determines the difficulty level involved. The software employs a content-based approach to identify spam and phishing emails by examining their contents. The Bayesian technique is utilized to categorize

spam words within the database. Future work will focus on expanding the phishing term collection and conducting a more extensive text mining analysis of email content. The use of artificial neural networks is also planned to achieve more precise findings and classification.

Peng et al. (2018) introduced an approach to identify phishing attacks by employing natural language processing and machine learning, specifically support vector machine (SVM). The study focuses on investigating metadata properties, such as email sender and receiver IDs, subject lines, email bodies, internet protocol addresses, and URLs embedded in phishing emails. Based on these desired properties, an anti-phishing algorithm is developed using SVM. Real-time success rates are achieved using datasets and library sets like accord.net. Future research may expand the SVM method to address a broader range of attacks, including CSS (Cross-Site Scripting) attacks.

Limitations: It is critical to recognize certain limitations when using machine learning for email-based anti-phishing techniques. These include a lack of labeled data, imbalanced datasets, difficulty adapting to new and evolving attacks, vulnerability to adversarial attacks, feature engineering and interpretability challenges, and the trade-off between false positives and false negatives. To overcome these constraints, ongoing research and development efforts are required to improve data collection, model training, feature selection, and interpretability. Integrating machine learning with other techniques, such as user behavior analysis and email authentication, can help anti-phishing systems perform better.

2.1.2 Deep learning based email phishing

Nguyen et al. (2018) presented a deep learning model with hierarchical Long Short-Term Memory (LSTM) and a supervised attention mechanism. The hierarchical LSTM structure is implemented first for words at the lower level, whose results are then passed to the LSTM structure in the sentences at the upper level to generate vector representation for the email. An attention mechanism is used to combine these two levels and to assign the contribution weights to each of the words and sentences in the email. A deep learning model is used to automate the feature engineering process for phishing email

2. Literature Review

Table 2.1: Summary: Email phishing using ML techniques

Author	Model / Algorithm	Features	Dataset(s)	Accuracy (%)	Limitations
Smadi et al. (2018)	Dynamic evolving Neural Networks	50 Hybrid	Phishing corpus & PhishTank: 4559 SpamAssasin:4559	Acc: 98.63 TPR:99.07 TNR:98.19	Limited comparison with existing methods and Lack of real-world validation
Islam and Abawajy (2013)	SVM, Adaboost, and NaiveBayes	21 Hybrid	SpamAssasin & Phishing corpus	Acc: 97	Complexity of analysis is high. High misclassification of test data
Khonji et al. (2012)	Lexical URL Analysis & RF	48 Hybrid	Phishing corpus: 4116, SpamAssasin: 4150	F-score: 99.37 FP: 0.59	Lacks the ability to accurately detect phishing emails with non-textual content in their body.
Gansterer and Pölz (2009)	SVM & J48	30 Hybrid	Phishing corpus: 5000, SpamAssasin: 5000	Acc: 97	Higher cost due to online features, classification speed depends on internet connection
Ramanathan and Wechsler (2012)	phishGILLNET	200 body	400,000 from multiple resource	Acc: 97.7	The complex architecture results in increased memory usage and computation time.
Ma et al. (2009)	DT, RF, MLP, NB, SVM	7 Hybrid	Phishing-46,525 Legitimate-613,048	Acc: 99	Did not utilize a verified dataset of phishing and legitimate emails for analysis.
Toolan and Carthy (2009)	Ensemble model (C5, Decision Tree, K-NN, LR, SVM, R-Boost)	5 Hybrid	SpamAssasin: 4202, Phishing corpus - 4563	F-Score: 99.31	May not handle real-world phishing emails accurately and small feature sets may hinder detecting complex phishing emails.
Abu-Nimeh et al. (2009)	LR, CART, SVM, NNET, BART,& RF	71 Hybrid	Phishing corpus:1403, Legit: 5152	Acc:97.09 (SVM)	Consumes more time and memory due to large set of features
Chandrasekaran et al. (2006)	SVM	25 structural features	In-House: Phishing-200 Legitimate-200	Acc: 95	Used very small dataset for analysis
Fette et al. (2007)	PILFER - RF & SVM	10 Hybrid	Phishing corpus: 860 SpamAssasin: 6950	Acc: 96	Resulted in 0.12% false positive and 7.35% false-negative rates. Achieved low performance with large datasets.
Alhogail and Alsabih (2021)	GCN	Body	CLAIR collection Phishing: 3685 Legit: 4894	Acc: 98.2	Used only body based features, accuracy is less compared to some existing works. Unknown No. of features.

detection. With the use of both the email headers and body, they achieved precision, recall, and F1 scores of 0.990, 0.992, and 0.991 respectively.

Castillo et al. (2020) proposed email threat detection using a distinctive neural network approach. The author described different approaches for detecting malicious content in emails. The proposed model is a combination of machine learning and natural language processing and used publicly available and private datasets. The model uses only email contents as input data set to classify emails as malicious or benign. In this work, the Gensim-Word2Vec model is used to generate numeric word vectors and achieved a testing accuracy of 95.68% with 1025 emails.

Hiransha et al. (2018) made use of IWSPA-AP 18 datasets to train the model consisting of Keras word embedding and Convolutional Neural Networks (CNN). The proposed model combining word embedding and CNN gives a vector representation for the words in the emails which are then used in the classification of legitimate and phishing emails. The proposed model achieved an accuracy of 96.8% using only email body text and 94.2% using both email header and body text.

Alhogail and Alsabih (2021) introduced a novel model for classifying phishing emails that leverages deep learning techniques, specifically Graph Convolutional Network (GCN) and natural language processing, to improve the accuracy of phishing detection using the body text of emails. The proposed model was tested using a supervised learning approach and demonstrated superior classification accuracy and performance compared to other deep learning methods. The training process was rapid, and the model achieved high true positive and true negative rates, as well as recall, precision, and overall accuracy. Text features in the email body represent a new research direction in the field of phishing detection since only a few studies have examined them. Moreover, the proposed classifier can effectively identify zero-day phishing attempts by identifying unseen body text, indicating its potential effectiveness in detecting the most recently disseminated zero-day phishing emails. Consequently, further studies are needed to evaluate its efficacy in detecting these zero-day phishing emails.

In recent research conducted by Li et al. (2020), LSTMs were employed for the clas-

sification of phishing emails. The primary dataset, which is privately owned, consisted of a large volume of two million emails. However, there are concerns regarding the methodology used to tag the training set. The labeling process involved a combination of k-means clustering and K-nearest neighbors (KNN) after a small sample had been manually labeled. Subsequently, features were extracted from both the email headers and content to expand the sample size and automate the labeling of the remaining unlabeled emails. The comparative analysis of labeling methods is also subject to scrutiny, as a different quantity of emails was labeled and analyzed to determine the accuracy of the proposed method compared to other labeling approaches.

Fang et al. (2019b) proposed a model called THEMIS, which is a combination of deep learning and Word2Vec techniques for phishing email detection. The model uses improved Recurrent Convolution Neural Networks (RCNN) with multilevel vectors and attention mechanisms. They used word level and character level vectorization for a rich set of vectors. The extracted vectors are tuned, trained, and tested using RCNN to obtain an efficient phishing accuracy of 99.848% and FPR of 0.043%. The obtained results are competitive, but the same would have been trained and tested with other WE and DL techniques. The author used multiple datasets to have a bulky dataset.

Bagui et al. (2019) proposed an approach that uses deep semantic analysis, ML, and DL techniques to classify phishing and legitimate emails. They used private datasets collected from various industries in the USA. The proposed approach uses hybrid features as text and achieved an accuracy of 98.89% with word phrasing and 96.34% without word phrasing using n-gram analysis and one-hot encoding techniques. In the proposed work, Bagui et al. (2019) claim that stop words are not removed and used both header and body features.

Ra et al. (2018) used the combination of word embedding, a neural bag of n-grams, and some deep learning models such as CNN, Recurrent Neural Network (RNN), LSTM, and MLP for the detection of phishing emails. Deep learning models are used to extract the optimal features and non-linear activation functions are used for classification. All the models are trained on an anti-phishing shared task corpus at IWSPA-AP 2018. The proposed model achieved a training accuracy of 99.1% with a word embedding vector

and LSTM network.

Verma et al. (2012) proposed "Detecting phishing emails the natural language way" in the year 2012, the first scheme used natural language processing techniques and contextual information in detecting phishing emails. The scheme uses all parts of the email including the header, text in the body, and the links present in an email. The proposed model named "PhishNet-NLP" operates between a mail transfer agent and a mail user agent. The proposed method achieves an accuracy of 97%. The obtained accuracy is comparatively less than in other works.

Gutierrez et al. (2018) proposed a model called SAF_E-PC for detecting a new form of phishing attacks, a semi-automated feature generation model for phishing classification. The model uses a huge corpus received from Purdue University's central IT organizations with the help of a state-of-the-art email filtering tool called Sophos installed on a Microsoft Exchange server. The author used three datasets as caught, uncaught, and benign of size 388,264 emails, 37,606, and 158,444 emails respectively, and used 806 features from email header, body, and links. The authors also tested their model with SpamAssassin open-source corpus and noticed the model performed better with collected real-time datasets. The authors claim that the proposed work is an extension of the work carried out by Verma et al. (2012). Used features in the proposed work are huge and may require more time to process in a real-time environment.

Verma et al. (2020) utilized natural language processing concepts to offer a comprehensive understanding of how phishing emails are classified. The classification of emails using natural language processing techniques includes evaluating the accuracy rate of various classifiers. The system produces a predicted matrix and classification report, along with the computed accuracy rates of classifiers. In future studies, raw and unstructured datasets will be employed for classification and clustering purposes.

Limitations: Anti-phishing techniques that utilize deep learning for email phishing have certain limitations. They face challenges in generalizing to novel attack patterns, are susceptible to adversarial manipulation, depend on acquiring ample and challenging to obtain training data, lack interpretability, encounter difficulties in contextual

2. Literature Review

Table 2.2: Summary: Email phishing detection using DL

Author	Model/ Algorithm	Algo-	Features	Dateset (s)	Accuracy (%)	Limitations
Bagui et al. (2019)	Deep Semantic Analysis, ML, DL, and Word Embedding		Hybrid - Body and Subject	Non-public	98.89	Used only text content without using sender info and attachments of an email, and Used only one-hot encoding for vector generation.
Nguyen et al. (2018)	NLP, DL, and H-LSTM		20 Hybrid	IWSPA-AP 2018	99.0	Need large amount of labeled data for training the DL models and achieved results with body and hybrid features are moderate.
Li et al. (2020)	LSTM, KNN, and K-Means		7 Header and body	Private	95.0	Used limited dataset, requires accurate labeling of phishing emails, and did not address issues of identity forgery and cloud attachments.
Castillo et al. (2020)	DNN, RNN, and Word2Vec	CN,	Body	Enron, APWG, and Non-public	95.68	Some unbalanced datasets used, reliance on pre-trained word embeddings, and only email contents used for phishing detection.
Ra et al. (2018)	CNN, LSTM, and WE	MLP,	Hybrid	IWSPA-AP 2018	99.1	Highly imbalance datasets are used to train the model, not tested the model with balanced datasets to justify the results.
Hiransha et al. (2018)	WE, CNN		Body	IWSPA-AP 2018	96.8	The dataset is highly imbalanced and leads to decrease in accuracy.
Harikrishnan et al. (2018)	Classical techniques, TF-IDF	ML	Hybrid	Combination of different publicly available datasets	90.29	Achieved accuracy is very low compared to all existing works. Over fitting due to unbalanced datasets
Valecha et al. (2021)	Word2Vec and Machine learning techniques.		Hybrid	Millersmile ³	96.52	Used persuasion cues, did not compare efficiency with other works, compared only with baseline model

comprehension, exhibit false positives and negatives, and demand substantial time and resources for training. It is necessary to update and expand the training data, enhance resilience against adversarial attacks, improve interpretability and contextual understanding, and integrate these techniques with other anti-phishing methods to bolster overall defenses.

2.1.3 URL Phishing detection using ML

Marchal et al. (2017) proposed a client-side application that extracts features mainly

from the URL and content of the website resulting in a 210 feature vector. The authors used a Gradient Boosting algorithm to classify phishing sites to achieve a significant detection rate. Using a large feature vector may include substantial time for the feature extraction and classification of URLs.

Li et al. (2019) proposed a stacking model combining Gradient Boosting Decision Tree, XGBoost, and LightGBM algorithms for detecting phishing web pages. The authors extracted features from the suspicious website's URL and Hypertext Markup Language (HTML). The extracted features contain 8 URLs and 12 HTML-based elements to generate a feature vector. The vector is fed to the stacked model for the classification and achieves an accuracy of 97.30%.

Jain and Gupta (2018b) proposed a client-side technique that uses features from the URL and source code of the suspicious site for classification. They applied five machine learning algorithms to identify the best classifier suitable for their dataset. RF had outperformed other classifiers with an accuracy of 99.09%.

El-Alfy (2017) proposed phishing websites based on probabilistic neural networks and clustering K-medoids. This framework combined unsupervised and supervised algorithms for training the nodes. K-medoid technology uses feature selection or transformation, and component analysis reduces space dimensionality. The technique achieved 96.79% accuracy by considering 30 features.

Zhang et al. (2014) proposed SMO for detecting and classifying Chinese phishing e-business websites. They used 15 unique and some generic domain-specific features to evaluate the model. They have used four different machine learning algorithms to classify phishing sites. Among all four algorithms, SMO performed the best in detecting phishing sites with an accuracy of 95.83%. The disadvantage of this approach is that it works better with Chinese websites only.

Rao and Pais (2019) proposed a new model for classifying phishing attacks, which utilizes heuristic features derived from URLs, source code, and third-party services. The aim was to overcome the limitations of existing anti-phishing methods. The model was tested using eight machine learning techniques, with the Random Forest (RF)

method yielding the highest accuracy rate of 99.31%. The author employed several orthogonal and oblique classifiers to identify the most efficient random forest classifier for detecting phishing websites. The principal component method proved to be the most effective oblique Random Forest (ORF) classifier, achieving an accuracy of 99.55%. Going forward, phishing attacks that involve embedded objects like Flash or HTML files will require the inclusion of additional heuristics. Other heuristics that exclude third-party services were explored to enhance the proposed model's effectiveness. The Proposed model will help reduce dependence on external services and minimize detection delays.

Natheztha et al. (2019) have proposed a three-phase attack detection system called the Web Crawler-based Phishing Attack Detector (WC-PAD), which can determine the occurrence of phishing attacks. The system uses various input factors, such as web traffic, content, and URLs, to classify websites as phishing or non-phishing. The effectiveness of the proposed system was validated by using datasets gathered from actual phishing situations used in an experimental study. It is found that the current methods for phishing detection are insufficient in addressing zero-day phishing website attacks. The proposed WC-PAD, however, demonstrated a high detection accuracy rate of 98.9% for both phishing and zero-day phishing attacks. Nonetheless, there is still room for improvement in terms of detection accuracy performance.

Pham et al. (2018) developed a neuro-fuzzy framework named Fi-NFN that utilizes aspects of web traffic and a URL to detect phishing websites. This framework is based on the innovative strategy of fog computing, as promoted by Cisco, that aims to create an anti-phishing model to monitor and protect fog users from phishing attacks in a discreet manner.

Yuan et al. (2018) proposed a method to detect phishing websites and their intended victims by analyzing features obtained from the URLs and website linkages. Various machine learning models are considered for phishing detection, the Deep Forest model has demonstrated superior performance and a high true positive rate. The proposed approach involves extracting features exclusively from the URLs and links on first-level web pages without accessing the content of second-level pages. This results in a fast and

accurate method for use in real-world situations. Additionally, a search operator-based approach for phishing target detection suggested, which also achieved a relatively high level of accuracy. However, further development of this feature in machine learning is necessary.

Chiew et al. (2019) introduced a new feature selection framework called Hybrid Ensemble Feature Selection (HEFS) for machine learning-based phishing detection systems. HEFS consists of two phases: the first phase involves using a novel Cumulative Distribution Function gradient (CDF-g) algorithm to generate primary feature subsets, which a data perturbation ensemble utilizes to create secondary feature subsets. The second phase of HEFS uses a function perturbation ensemble to obtain baseline features from the secondary feature subsets. The experiments showed that combining HEFS with the Random Forest classifier resulted in the most effective detection, accurately identifying 94.6% of phishing attempts. However, further improvements to the detection rate will require enhancing the accuracy of the Random Forest classifier.

Yadollahi et al. (2019) have developed a resilient detection system that can adapt to the surrounding environment and phishing websites. This system utilizes machine learning to differentiate between legitimate and phishing websites online and incorporates many features. The proposed approach is a completely client-side solution that extracts various types of discriminative information from URLs and web page source code, eliminating the need for assistance from a third party. However, while this approach effectively detects various types of phishing websites, it may not be able to identify zero-day attacks.

Zhang et al. (2017) propose a novel framework for detecting phishing web pages that incorporates textual content labels as part of its features. This framework comprises rule-based, URL-based, web-based, and text-based features selected using an effective two-stage Extreme Learning Machine (ELM). In the first stage, ELM is used to build classification models for the textual content of web pages, with Optical Character Recognition (OCR) utilized as an aid tool to extract text from web pages with picture formats. In the second stage, a classification model on hybrid features was created using a linear combination model-based ensemble ELMs (LC-ELMs), with weights

determined using the generalized inverse. The framework's future development plans include an incremental approach to upgrading the detection model and improving the textual content categorization process.

Ghimire et al. (2021) presented different approaches for detecting phishing URLs based on feature extraction using machine learning. Network-based and URL-based features fed into the machine learning classifiers. The proposed system is categorized into five parts: data collection, feature engineering, data preprocessing, algorithm classification, and performance evaluation. However, there is a need to improve the accuracy of phishing detection to enhance the system's overall performance.

Nagunwa et al. (2019) proposed a new framework for anticipating zero-hour phishing websites by introducing hybrid features demonstrating good prediction performance. The prediction performance of the features explored using eight machine-learning techniques. The Random Forest approach showed the best performance, with an accuracy of 98.45% and a false negative rate of 0.73%. To further enhance prediction performance and efficiency beyond existing works, additional research should focus on exploring novel perspective features and utilizing advanced machine learning methodologies, such as deep learning and online learning.

In their study, Adewole et al. (2019) presented a novel hybrid rule induction algorithm aimed at distinguishing between legitimate and phishing websites. They also explored the possibility of early detection of phishing attacks through a combination of rules generated by two commonly used rule induction algorithms, namely JRip and PART. The PART algorithm proved to be more effective in detecting phishing than JRip. To address zero-day phishing attacks, future research should focus on investigating the use of adaptive machine-learning techniques for phishing detection.

The study by Afek et al. (2017) focused on analyzing the detection of malicious URLs, considering two datasets, namely Phish Tank and UCI. The researchers first collected the most effective features from these datasets using a DBA-based detector module. They then generated seventy-eight optimal rules using association rule mining based on these features. However, repetitive elements in the rules may limit the effec-

tiveness of identifying fraudulent URLs. The study suggests that identified problem can be solved using the Frequent Rule Reduction technique and a classification approach to predict phishing websites.

Kumar and Indrani (2021) conducted a study comparing manual feature selection methods with those used in the Filter method and proposed a new manual feature selection methodology. The study also compared the performance of various machine learning classification methods, such as Naive Bayes, J48, and HNB, to efficiently detect phishing websites. The researchers closely monitored the performance of a classifier that combined HNB and J48 to address the identified issue. However, given the continuous evolution of new phishing techniques, there is a need to improve the feature selection process to ensure effective detection.

Zabihimayvan and Doran (2019) introduced the Fuzzy Rough Set (FRS) theory that relies on an anti-phishing approach. The theory allows the selection of the most beneficial features from three established data sets. These features fed into three popular classifiers for detecting phishing attempts. To evaluate the effectiveness of the Fuzzy Rough Set feature selection in creating a comprehensive phishing detection, the classifiers trained on an independent out-of-sample data set.

Alam et al. (2020) proposed a model to detect phishing attacks utilizing machine learning algorithms: random forest and Decision Tree (DT). The researchers employed a Kaggle dataset of phishing attacks to support ML processing. Utilized feature selection methods such as principal component analysis to examine the dataset's characteristics. DT was used to categorize websites, while RF was employed for classification. RF tackled the over-fitting issue resulting in 97% accuracy. Nonetheless, building a phishing attack detection method based on CNN is necessary.

The study by Garcés et al. (2019) explored various machine-learning approaches to identify abnormal behavior linked to phishing web attacks. The detection of phishing attacks is achieved by analyzing URLs and determining their trustworthiness based on specific characteristics. The analysis involves using contaminated data sets and Python tools to generate real-time information, which can aid in making proactive decisions

to reduce the impact of such attacks. Nevertheless, it should be noted that machine learning methods are not always foolproof. By acknowledging these limitations, the community can work towards developing new tools to enhance these imperfect techniques' effectiveness.

Jain and Gupta (2018a) introduced a machine learning-based solution named PHISH-SAFE to combat phishing attacks, which leverages URLs as a key feature. Specifically, they utilized 14 URL elements to determine whether a website is malicious. To evaluate the efficacy of the proposed system, the researchers used more than 33,000 authentic and phishing URLs to train the system with Support Vector Machine (SVM) and Naive Bayes classifiers. The results revealed that the system achieved over 90% accuracy in detecting phishing attacks when an SVM classifier was employed. Future improvements to the system could include adding more features to increase its accuracy. Furthermore, alternative machine learning techniques could be explored to enhance the solution's effectiveness.

Kunju et al. (2019) presented a concise overview of various machine learning approaches to identify phishing websites, such as KNN Algorithm, Naive Bayes, Decision Tree, Support Vector Machines, Neural Networks, and Random Forest algorithms. The authors emphasized the importance of raising awareness about social attacks and their detection, as many users remain oblivious to this serious threat and continue to fall victim to it. Despite the potential consequences of sharing sensitive information on hidden phishing websites, many consumers are unaware of this issue and willingly submit their personal data. Thus, detecting and preventing phishing attacks remains a significant challenge for future research and development.

Rashid et al. (2020) suggested an efficient approach to detecting phishing attacks using machine learning. The study showed that the proposed method achieves optimal performance by utilizing only 22.5% of the new capability and combining it with the Support Vector Machine (SVM) classifier, resulting in the reliable identification of 95.66% of phishing and legitimate websites. The authors suggest that future research could examine the impact of feature selection with other classification algorithms to further improve the system's effectiveness.

Sahingoz et al. (2019) proposed a real-time anti-phishing system incorporating seven distinct categorization algorithms and features based on NLP. This system differs from previous studies in various ways: it is not language-dependent, uses significant amounts of both genuine and phishing data, operates in real-time, identifies new websites, does not rely on third-party services, and employs classifiers with a plethora of features. To evaluate the system's performance, the researchers created a new dataset and used it to assess the experiment's findings. The study results and the comparison of the various classification methods indicate that the Random Forest approach, which only uses NLP-based features, is the most effective, with a 97.98% accuracy rate in detecting phishing URLs. If a website is identified as a phishing site, it can be added to the network's local blacklist and prevented from receiving further requests.

Salihovic et al. (2018) employed machine-learning techniques to eliminate the human factor in security breaches. The authors used two datasets, the Phishing Websites Data Set from UCI and the Spam Emails Dataset, along with Weka software, to train and test six successful algorithms, including Random Forest, k-Nearest Neighbor, Artificial Neural Network, Support Vector Machine, Logistic Regression, and Naive Bayes. The outcomes revealed that the Random Forest algorithm was the most effective for both datasets. These results serve as a foundation for future initiatives to detect online fraud more quickly and accurately.

Patil et al. (2018) discussed three techniques for detecting phishing websites. The first technique involves scrutinizing various URL components, while the second involves verifying the website's authenticity by investigating its location and ownership. The third technique involves visually examining the website to determine its legitimacy. These techniques rely on machine learning algorithms and techniques to assess various attributes of the URL and the website. However, the system's drawback is that it may produce a few false positive and false negative results. There is a need to enhance the machine learning algorithm's performance by incorporating richer features to overcome this limitation. This will lead to a significant improvement in accuracy and help to avoid these limitations.

Buber et al. (2017) have developed a technique for identifying URLs used in phish-

ing attacks by utilizing NLP approaches and removing certain aspects of the proposed system. The extracted features are analyzed by two groups, one focusing on distinguishing between malicious and legitimate URLs using individual characteristics. At the same time, the other simply vectorizes the URLs words and analyzes their usage. The experimental investigation includes three separate test scenarios, namely, Word Vectors, NLP-based features, and a hybrid approach combining both features. The experiments use Random Forest, Sequential Minimal Optimization (a kernel-based strategy), and Naive Bayes (a statistical-based approach) algorithms. The Random Forest algorithm achieves the highest success rate at 97.2%, outperforming the other algorithms evaluated. However, there is still a need to improve the performance of tree-based, kernel-based, and statistical-based algorithms to detect phishing attacks with greater accuracy.

Balamurugan and Jayabharathy (2022) conducted a study that concentrated on predicting multimedia bullying content, encompassing text, images, video, and audio. They used machine learning techniques such as K-nearest neighbor, support vector machine, naive Bayes, and random forest to forecast bullying in multimedia content. Textual bullying harms social media, and it is crucial to identify and classify social media interactions accurately as bullying. When users reveal personal information on unrelated websites, it could result in phishing attacks, and informing users about malicious attacks becomes more challenging. The researchers proposed a technique that uses SVM text categorization to determine whether the text in social networks involves bullying. Furthermore, modern techniques are necessary to improve the detection of phishing.

Limitations: Tables 2.3 and 2.4, summarizes the reviews of phishing URL detection using machine learning techniques. Despite the advantages of this technique, such as its ability to identify fraudulent activities, there are also several drawbacks. For instance, the method may not always be entirely accurate. It may detect phishing attacks with less precision, not evaluate the impact of feature selection, and not detect online fraud quickly and accurately. Given these limitations, there is a need to enhance the performance and efficiency of machine learning techniques.

Existing web-based anti-phishing techniques using machine learning models have

Table 2.3: Summary: URL based phishing detection using ML

Author(s)	Technique	Features	Datasets	Significance	Limitations
Alam et al. (2020)	RF & DT	32	Kaggle	Attributes selection using principal component analysis (PCA) and achieves 97% accuracy using RF algorithm.	Depends on a single feature selection method without justification and used limited evaluation metrics.
Garcés et al. (2019)	Machine learning	URLs	Kaggle (420464 URLs)	provide real-time information and minimize the impact of an attack. ML techniques detect in Latin American and propose cognitive security architecture	High false positives or false negatives in the classification process. Did not address dynamic nature of phishing techniques.
Jain and Gupta (2018a)	PHISH-SAFE (SVM & Naive Bayes)	14	Phishtank.com (32951)	PHISH-SAFE: URL Phishing Detection System Using Machine Learning.	Anti-phishing technique with only 90% accuracy which is comparatively less.
Rashid et al. (2020)	SVM classifier	5	UCI repository	Performs best for reliably identifying 95.66% of phishing and suitable websites using only 22.5% of innovative functionality.	The study used only one performance metric i.e. accuracy for model evaluation and includes small dataset size.
Sahingoz et al. (2019)	RF with NLP based features	278	PhishTank (37175) & Legitimate: Yandex search API (36400)	It is language independent, uses massive amounts of legitimate and phishing data, executes in real-time, can detect new websites, independent of third-party services, and employs feature-rich classifiers	Dataset creation biases, class imbalance and huge feature set focusing on NLP.
Salihovic et al. (2018)	ML algorithm and Ranker + PCO	-	UCI repository	Eliminate the human element from security breaches carried out, achieved accuracy is of 97.33%	Limited dataset, limited algorithm selection without justification, lack of explanation for experimental setup and absence of statistical significance analysis
Patil et al. (2018)	ML technique and algorithm	12	-	Proposed a hybrid solution for detecting and preventing phishing websites using machine learning approaches. Evaluate various URL and website attributes	Limited model testing approaches and used only one dataset evaluation. The study has minimal false positive and false negative results.
Buber et al. (2017)	NLP & ML	40	PhishTank & Yandex search API	By employing NLP, the system generates certain features and applies three distinct ML techniques to classify the URLs. Improved 7% performance from their previous work.	Limited dataset (3717 malicious and 3640 legitimate URLs)

Table 2.4: Summary: Feature extraction based Phishing URL detection using ML.

Author(s)	Technique	Features	Datasets	Significance	Limitations
Zabihimayvan and Doran (2019)	Fuzzy Set (FRS) ML	24, 30 & 9	UCI, Mendeley, & UCI2	FRS theory tool used to select the most effective features, the features are trained using ML algorithms and gained f-measure of 95% with RF.	The study evaluates the efficacy of FRS feature selection on three benchmarked datasets; the performance may not be optimal for all phishing attacks.
Yuan et al. (2018)	Deep Forest model	12	PhishTank (2892) & Alexa Ranking and Network security challenge repository (3305)	Competitive performance by identifying each website in less than one second using statistical features and linguistic features of URLs and linkages of webpages.	Reliance on third-party services, need to access webpage content for content-based approaches, used small size dataset for testing. It may not be effective against new and evolving phishing techniques.
Chiew et al. (2019)	ML-based Hybrid Ensemble Feature Selection (HEFS)	Baseline (10) & all (48)	UCI	A new feature selection framework called HEFS performs best when integrated with an RF classifier, where the baseline features correctly distinguish 94.6% of phishing and legitimate websites	Features used to detect phishing are specific to certain attacks and may not be able to detect new and unknown phishing attacks.
Yadollahi et al. (2019)	Machine Learning	38	Private - Phishing (3983), Legitimate (4021)	Online and feature-rich ML technique designed to differentiate between phishing and legitimate websites. Extracts various kinds of discriminative information from URLs and web page's source code.	May face challenges in detecting sophisticated phishing attacks that use advanced obfuscation techniques. Limits scalability due to unavailability of third-party service. Unable to identify zero-day attacks.
Zhang et al. (2017)	Extreme Learning Machine (ELM)	14	DS-1: (PhishTank (2784), Statscrop (3121)), DS-2: (SMS & Email (500), Search engine & Statscrop (500))	Two-stage ELM approach using URL, HTML, source code, and web hybrid features to detect phishing web pages. Found effective in English and Chinese phishing web pages.	Framework is evaluated only on English and Chinese phishing web pages, did not consider dynamic nature of phishing attacks.
Ghimire et al. (2021)	Machine learning	21	PhishTank (450176)	Various ML algorithms are used to classify phishing URLs; the model performance is evaluated on original, under-sampled, and over-sampled data.	Relies on network-based and length-based features limit the detection capability and use default hyperparameters without fine-tuning for better accuracy.
Nagunwa et al. (2019)	Eight machine learning	31	PhishTank (9019) & Google and Bing search engines (1733)	Proposed framework for predicting zero-hour phishing websites by introducing new hybrid features with eight ML algorithms. RF achieved accuracy and false negative rates of 98.45% and 0.73%.	Need for continuous feature discovery, the scalability and efficiency of the approach for larger-scale deployments not extensively discussed, exploring new potential features and online learning for further improvement
Adewole et al. (2019)	Hybrid rule-based model	DS1 - 10 & DS2 - 30	UCI (DS1: (548), Phishing (702) & Suspicious (103) - Total (1353), DS2: Phishing (4898) & Legitimate (6157))	The model combines the strengths of JRip and PART algorithms, to improve the accuracy of phishing URL detection, the model uses a comprehensive set of features to detect phishing URLs	The approach is computationally expensive when working with large datasets because it utilizes two rule induction algorithms, and its effectiveness against new or unseen phishing techniques may be limited since it relies on a fixed set of features and algorithms.

limitations, including a lack of sufficient training data, vulnerability to adversarial attacks, difficulty identifying zero-day attacks, challenges with imbalanced data, struggles in generalizing diverse phishing attacks, and privacy concerns. To overcome these limitations, ongoing research and development are needed to improve model performance and robustness. Combining machine learning with other techniques like user education and regular security updates can strengthen anti-phishing defenses.

2.1.4 URL Phishing detection using Deep Learning

Bahnsen et al. (2017) Bahnsen et al. (2017) used a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) to classify phishing URLs in their work. The authors used 3-fold cross-validation to compare the traditional Random Forest (RF) machine learning algorithm to the LSTM-based approach. The RF algorithm used 14 features for URL statistical and lexical analysis and achieved a 93.5% accuracy. The RNN model, which processed the URLs directly, outperformed the RF algorithm with an accuracy of 98.7%. Notably, the RNN approach eliminated the need for time-consuming and labor-intensive manual feature extraction.

Le et al. (2018) use a deep learning model to detect phishing URLs. They use the URLNet framework to learn a nonlinear URL embedding for malicious URL detection directly from the URL. To learn URL embedding, URLNet uses CNN specifically for both characters as well as words of the URL string. The proposed method has similar accuracy for word and character levels and performs much better than other methods. This method may fail if the phishing sites are represented with short URLs (bitly, goo, tiny, etc.) and data URLs.

Zhao et al. (2018) proposed a Gated RNN model and showed that Gated Recurrent Unit (GRU) outperforms the RF classifier with 21 features and achieved 2.1% better efficiency than RF, i.e., 98.5%. But, here, only URLs are used as data sets and need to transform all characters into vectors to learn hidden patterns. Hence, GRU needs more time to train and requires system architecture to be optimized for better performance.

Mohammad et al. (2014) proposed a model to predict phishing sites based on self-structuring neural networks. They used 17 features extracted from the URL and source

code of the website. These features are used to classify websites in artificial neural networks. This model should be regularly retrained with up-to-date training data sets.

Feng et al. (2018) proposed a novel classification model for detecting a given website's legitimacy. They used the Monte Carlo algorithm (Zhou et al. 2016) for training the model and the risk minimization principle to avoid overfitting in the proposed model. They have adopted 30 features from the UCI4 repository and could achieve an accuracy of 97.71%.

Yi et al. (2018) proposed a deep learning framework with two feature sets: original and interaction features. The original features extracted from the URL analysis, i.e., the presence of special characters (@, -, Unicode), count of dots, and domain age. The interaction features extracted from the website's source code, i.e., in-degree, out-degree, frequency of accessing URL, and cookie absence. Deep Belief Network (DBN) is applied to the extracted features and achieved an accuracy of 90% true positive rate and 0.6% false positive rate.

Saha et al. (2020) have introduced a data-driven approach to leverage deep learning techniques for detecting phishing websites. Specifically, their method employs a multilayer perceptron, also called a feed-forward neural network, to predict phishing web pages. The dataset used in their study was obtained from Kaggle and contained information from 10,000 websites. The approach yielded promising results, with 95% accuracy during training and 93% during testing. Notably, the model can effectively recognize unknown web pages and learn from the dataset, as evidenced by the small gap between training and test accuracy. The accuracy of their authentic website detection method is higher than that of the current phishing detection system, at 98.4%. Future research will explore using more layers in neural networks and more precise models, such as backpropagation neural networks, for detecting phishing attacks.

A deep learning-based end-to-end automatic phishing web page classification method called HTML Phish was proposed by Opara et al. (2020). In this approach, the HTML content of a web page is fed into HTML Phish, which employs convolutional neural networks to create an optimized network to learn the semantic connections between

characters and words in the HTML document. Additionally, convolutions are applied to a matrix concatenation of character and word embeddings to ensure that new words in test HTML documents are actively incorporated. In the future, this model will be compared with feature engineering-based models that only extract features from HTML pages.

Maurya and Jain (2020) proposed an anti-phishing architecture that internet security providers (ISPs) could use to combat phishing attacks on their level and provide secure connections to end-users, irrespective of their device configurations. The proposed architecture employs deep learning categorization as the backend to detect phishing websites at the end of the ISP. By introducing an intermediate security layer at the ISPs between multiple servers and end-users, this method adds an additional layer of security. With a single point of blocking, millions of users can be protected against a specific phishing attempt, making implementing this system highly effective. End-users receive secure services regardless of their system configurations, without the need for powerful computing devices, as the computational overhead for phishing detection models is borne solely by the ISPs. Future research will focus on developing an adaptive mechanism that can handle DNS cache poisoning of end-user systems. Furthermore, there is a need to improve the accuracy of the prediction model by utilizing more adaptive optimization approaches.

Adebowale et al. (2023) designed and implemented a deep learning-based phishing detection system that utilizes website content, including photos, text, frames, and the universal resource locator. The resulting Intelligent Phishing Detection System (IPDS) is a hybrid classification model that combines the long short-term memory algorithm and convolutional neural network. A comprehensive experimental investigation was conducted to evaluate the effectiveness of IPDS in identifying phishing web pages and phishing attacks on big datasets. The LSTM and CNN combination is a deep learning technique that combines pictures, text, and frame information to produce a unified phishing detection scheme. In the future, the focus will be on improving the scheme's accuracy and developing a web browser plugin based on a deep learning algorithm to recognize web phishing across platforms and provide real-time consumer protection.

Singh et al. (2020) developed a phishing detection system that utilizes deep learning techniques to detect such attacks. The system uses convolutional neural networks to analyze URLs and identify phishing websites in real time. URL concatenation was performed to prepare the dataset for the learning algorithm due to the large number of phishing and legitimate website URLs in the dataset. The primary goal of this system is to differentiate between legitimate and phishing URLs, but it achieves only 98% accuracy in detecting phishing attacks. Therefore, there is a need to enhance the performance of deep learning techniques to detect phishing attacks more efficiently and accurately.

Yang et al. (2019) proposed a rapid multidimensional feature phishing detection method based on deep learning. In the first phase, character sequence features of the provided URL are extracted and used for rapid classification by deep learning without any external assistance or prior knowledge of phishing. In the second phase, the multidimensional features are created by combining deep learning's rapid classification output with URL statistical data, webpage code features, and webpage text features. This method can reduce the time required to detect potential phishing attacks before triggering a threshold. The accuracy of the proposed method reaches 98.99%, with a false positive rate of only 0.59%, using a dataset comprising millions of legitimate and phishing URLs. Future research can further improve the efficiency of detection.

Yerima and Alzaylaee (2020) proposed an approach based on deep learning to achieve highly accurate identification of phishing sites. NLP is used to extract features from the URLs of these sites. Convolutional neural networks are utilized in this method for high-accuracy classification, distinguishing between legitimate and phishing websites. The model was evaluated using a 6,157 legitimate and 4,898 phishing websites dataset. Numerous experiments reveal that CNN-based algorithms effectively identify unknown phishing sites. The CNN-based technique outperformed conventional machine learning classifiers tested on the same dataset, with a phishing detection rate of 98.2% and an F1-score of 0.976. This study's technique surpasses the most recent deep learning-based phishing website detection methods. As a future work, the model training process can be improved by automatically identifying and selecting the most

influential factors that, when combined, produce the best-performing CNN model.

In their study, Zinovyeva et al. (2020) explored the possibility of utilizing deep machine learning and natural language processing for autonomous content monitoring. The researchers synthesized previous studies on identifying antisocial behavior online and compared relevant methodologies with modern NLP models. They discussed important NLP methodological advancements such as bidirectional encoding, attention, hierarchical text representations, and pre-trained transformer-based language models. Additionally, they introduced a pseudo-sentence hierarchical attention network as an extension of previous approaches.

In their research, Jonker et al. (2021) investigated the use of natural language processing, a subsection of Machine Learning, to resolve the issue of phishing. They also examined various ML techniques, including RNN, LSTM, CNN, TD-IDF, and multiple NLP techniques, such as Word2Vec, Doc2Vec, and BERT. All of these techniques generated precise classification results, with f1-scores ranging from 90.03% to 98.94%. Nonetheless, there remains necessary to enhance the performance of detecting phishing attacks.

A summary of the related works using deep learning to classify phishing URLs is tabulated with six metrics in Table 2.6. These metrics include the detection of phishing sites that replace textual content with an image (Image-based phishing), the detection of phishing sites that contain most of the hyperlinks directed towards a common page (Common Page Detection), the detection of phishing sites that are hosted in any language (language independence). The detection of phishing sites consists of a maximum number of broken links (Broken links), detection of phishing sites based on different models, and the number of features used to classify phishing sites.

Limitations: Table 2.5 and 2.6 provides an overview of deep learning methods for detecting phishing attacks. While deep learning has shown promise in detecting phishing, there are some limitations, including difficulties in determining website legitimacy, addressing only common issues, lack of comparison with engineering-based models, limitations in detecting multidimensional features on websites, the poor performance

Table 2.5: Summary: Anti-phishing techniques based on Website phishing using DL

Author(s)	Technique	Features	Datasets	Significance	Limitations
Saha et al. (2020)	Data-driven	10	Kaggle (10000 web pages)	A data-driven framework based on deep learning was proposed to detect phishing websites, aiming to surpass the reliance of traditional methods on digital platforms.	Achieved comparatively low training and testing accuracy. The performance may be improved by adding more neural network layers and back propagation networks.
Opara et al. (2020)	HTMLPhish a DL Technique	500000 HTML documents	In-house generation using web crawler and Beautiful Soup library	Learn the semantic connections between the characters and words in the HTML document convolutions were applied to a concatenation of the character and WE matrix	Absence of thorough comparison with feature engineering based models and real time browser extension implementation is pending.
Maurya and Jain (2020)	Anti-phishing architecture using DL	30	UCI repository (11055 websites)	Anti-phishing framework based on deep learning at ISPs level to ensure safety. It adds an intermediate security layer at ISPs and is placed between numerous servers and end-users ensuring single point of blocking.	Limited focus on real-time phishing detection, lack of empirical evaluation and lack of detail on adaptive optimization techniques and the phishing detection models is limited only to ISPs.
Adebowale et al. (2023)	IPDS with LSTM and CNN	35	PhishTank & Common crawl(1 million URLs and 10000 images)	The study explore differentiating unique legitimate URLs from the phishing URLs using LSTM and CNN in combination with IPDS classifier and achieved an accuracy of 93.28%	Dataset bias or imbalance may affect performance and generalizability. Insufficient evaluation metrics used and limited scope of features.
Singh et al. (2020)	CNN	Direct URLs	PhishTank (37175) & Yandex Search API (36000)	Traditional methods of detecting phishing are not always effective; used CNN a deep learning technique that extracts features directly from the URLs. It avoids feature engineering.	The achieved accuracy showed a slight improvement compared to their previous work; however, it remains relatively low when compared to other existing studies.
Yang et al. (2019)	CNN-LSTM Hybrid network	20	URL phishbank.com (1021758) & dmoz-24 Web page tools.net (989021)	The system employs a dynamic category decision algorithm for quick detection without prior phishing knowledge, as well as multidimensional feature detection for accurate results.	Lack of implementation to extract webpage code and webpage text features.
Yerima and Alzaylaee (2020)	CNN	30	UCI repository (Phishing - 4898, Legitimate - 6157)	presents a deep learning-based approach to enable high accuracy detection of phishing sites, reaching 98.2% phishing detection rate with an F1-score of 0.976.	The computational complexity may limit its scalability to larger datasets and may not be effective against sophisticated and targeted phishing attacks designed to evade detection
Zinovyeva et al. (2020)	DL and NLP	URLs	Wikipedia, Twitter, Facebook, & Formspring	The study investigates the potential of automatic content monitoring using deep machine learning and NLP, incorporating NLP advancements and introducing a new pseudo-sentence hierarchical attention network.	Computationally expensive model, hierarchical structures may not always enhance antisocial online behavior detection accuracy.

Table 2.6: Summary of DL based URL phishing detection using six metrics.

Techniques	Image based Phishing	Common page based Phishing	Language independence	Broken Links	Models	Features.
Yao et al. (2018)	No	No	Yes	No	Neural networks	17
Parsons et al. (2019)	Yes	Yes	Yes	No	PNN K-Modoids Clustering	30
Al-Musib et al. (2021)	No	No	Yes	No	Gated Recurrent Neural Network	Direct URLs
Balim and Gunal (2019)	No	No	Yes	No	Convolution Neural Network	Direct URLs
Jia et al. (2021)	No	No	Yes	No	Recurrent Neural Network	Direct URLs
Yang et al. (2019)	No	No	Yes	No	CNN-LSTM Hybrid Network	Direct URLs
Mondal et al. (2022)	Yes	Yes	Yes	No	Neural Network	30
Sonowal (2022a)	Yes	No	Yes	Yes	Deep Learning DBN	8

of sentimental classifiers, and detecting only voice spoofing attacks. Therefore, there is a need to improve deep learning techniques to enhance their ability to detect phishing attacks in both email and website domains.

Despite current technological advancements, website or URL phishing still faces limitations. These include evolving phishing techniques, polymorphic attacks, challenges posed by HTTPS encryption, zero-day attacks, phishing in legitimate domains, and the use of shortened URLs and redirectors. A multi-layered approach integrating URL analysis, content inspection, behavior monitoring, machine learning techniques, and user education is necessary to mitigate these limitations. Continuous research and development are vital to avoid emerging phishing techniques and enhance the detection and prevention of website and URL phishing attacks.

2.2 EVALUATION METRICS

Evaluation metrics are used to compare different models and algorithms to determine which is most effective for a given task. Evaluation metrics can be used to assess the accuracy and performance of a model, such as its ability to generalize to unseen data and

optimize a model's hyperparameters, such as learning rate, to ensure it is as accurate as possible. And also can be used to select the most effective features for training a model, such as those with the highest correlation with the target variable, interpret a model's output. In the subsequent chapters of the thesis, various tabulated results to compare existing works. We have used traditional metrics to evaluate the performance of our proposed systems. The evaluation metrics used to evaluate our proposed models given below.

- The sensitivity or recall is known as true positive rate (TPR):

$$TPR = \frac{TP}{(TP + FN)} * 100 \quad (2.1)$$

where, $TP =$ No. of phishing data classified as phishing, and $(TP + FN) =$ Total no. of phishing data.

- Specificity as true negative rate (TNR):

$$TNR = \frac{TN}{(TN + FP)} * 100 \quad (2.2)$$

where, $TN =$ No. of ham data classified as ham, and $(TN + FP) =$ Total no. of ham data.

- Accuracy (Acc):

$$Acc = \frac{(TP + TN)}{(TP + FP + TN + FN)} * 100 \quad (2.3)$$

where, $(TP + TN) =$ No. of correctly classified phishing and ham data, and $(TP + FP + TN + FN) =$ Total no. of records in the dataset.

- Precision (P):

$$P = \frac{TP}{(TP + FP)} * 100 \quad (2.4)$$

where, $TP =$ No of phishing input data classified as phishing, and $(TP + FP) =$ Total no. of input data samples classified as phishing.

- F-score (F) :

$$F = 2 * \frac{P * TPR}{P + TPR} \quad (2.5)$$

- Matthews Correlation Coefficient (MCC): This measure is considered as a balanced measure, used for different class size datasets. MCC provides a correlation coefficient between predicted and observed outcomes.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (2.6)$$

Equations (2.1) through (2.6) evaluate the input sample based on their positive and negative rates. TP, or truly positive, represents the number of phishing samples correctly classified as phishing, while TN, or truly negative, represents the number of legitimate samples correctly classified as legitimate. FP, or false positive, is the number of legitimate samples incorrectly classified as phishing, while FN, or false negative, is the number of phishing samples incorrectly classified as legitimate. These basic metrics are used to calculate recall (2.1), specificity (2.2), accuracy (2.3), precision (2.4), F-measure (2.5), and Matthews Correlation Coefficient (2.6), as shown in Table 2.7.

Table 2.7: Confusion matrix

n=M	Predicted YES	Predicted NO
Actual YES	TP	FN
Actual NO	FP	TN

2.3 DATASETS USED

The thesis focuses on the classification of email and website phishing. In the research on email phishing, two publicly available open source datasets from separate repositories were utilized. The ham emails were exclusively collected from the SpamAssassin.com⁴ web repository, while the phishing emails were sourced from the phishing corpus monkey.com⁵. In addition to these open source datasets, in-house datasets were created to address issues such as redundancy and period mismatch that were observed in the open source datasets. Regarding the website phishing research, the dataset was obtained from Rao and Pais (2019), and a minimal set of features was selected from this dataset. The subsequent procedures for preparing the email dataset are discussed in detail below.

2.3.1 Dataset preparation

Two methods were used to create the Datasets: 1. Open-source corpus and 2. In-house corpus generation.

⁴<https://spamassassin.apache.org/old/publiccorpus/>

⁵<https://monkey.org/jose/phishing/>

2.3.1.1 Open-source corpus

Data preparation is a significant aspect of the current research. Prior studies typically relied on two open-source datasets (SpamAssassin and monkey.com) consisting of phishing and legitimate emails. However, these datasets contained duplicate emails from previous years. To address this, Dataset-1 was created by selecting 6295 unique legitimate emails and 9135 phishing emails after removing duplicates. Subsequently, Dataset-2 was formed by combining the phishing emails from Dataset-1 with 18270 legitimate emails from an in-house corpus.

2.3.1.2 In-House corpus

One of the most vital steps in email phishing is dataset creation by investigating individual emails. To be apprised of the daily phisher activities, researchers must have updated, new, real-time data. Most research has utilized existing open-source datasets such as Dataset-1, shown in Table 2.8. The selected SpamAssassin datasets were collected in 2002, and the Phishing corpus datasets were collected between 2004 to 2007 and from 2015 to 2017. The ham email datasets were older than the phishing corpus emails collected after November 2004. Since these open-source datasets do not match their period, the period mismatch may fail phishing email detection. Phishers can alter insignificant parameters to trap victims. The phisher's behaviors and strategies are changing every day to fool victims by acquiring sensitive information to defraud users. To combat this problem and deal with the current tricks, Dataset-3 was created using real-time internal phishing and legitimate datasets. The new repositories were collected from institution students, research scholars, relatives, and friends to comprehend the behavior of fraudsters with a varied set of users. The chosen emails were analyzed manually and labeled as phishing or legitimate. The selected datasets and their sizes are tabulated in Table 2.8. The following steps are taken to distinguish emails as either phishing or legitimate during the formation of datasets:

- Analyse the behavior of suspicious emails.
- Analyse the source code of the original email message.

Table 2.8: Datasets used

Dataset	Legitimate Emails	Phishing Emails	Total
Dataset-1	6295	9135	15430
Dataset-2	18270	9135	27405
Dataset-3	18270	8986	27256

- Analyse the Google warning indicators.
- Using MxTOOLBOX⁶ online tool to analyze email headers.

2.4 SUMMARY

In this chapter, we provided a comprehensive overview of various studies related to phishing. We explored the application of anti-phishing techniques and discussed using different methodologies such as machine learning, deep learning, word embedding, natural language processing, and feature extraction for detecting phishing attacks. Furthermore, we delved into the evaluation metrics employed in these studies, the datasets utilized, and the methods employed for dataset preparation. By examining these essential aspects, we better understood the diverse approaches and strategies used in phishing detection.

⁶<https://mxtoolbox.com/Public/Tools/>

CHAPTER 3

PHISHING EMAIL DETECTION FRAMEWORK USING WORD EMBEDDING AND MACHINE LEARNING

Phishing email detection is crucial for maintaining online security, as these deceptive messages aim to deceive recipients and extract sensitive information. Researchers have developed techniques using word embedding and machine learning to combat this threat. Word embedding represents text as numerical vectors, capturing contextual and semantic information to improve detection accuracy. Machine learning algorithms analyze and classify emails based on labeled datasets, learning patterns to distinguish phishing from legitimate emails. Our proposed framework integrates word embedding techniques (Word2Vec, FastText, TF-IDF) with machine learning algorithms (decision trees, random forests, support vector machines, logistic regression, and XG boost) for comprehensive phishing email detection. Transforming email content into numerical representations enhances feature extraction, while training models on large labeled datasets ensures accurate detection. Our goal is to create an efficient and robust system that effectively identifies and mitigates phishing attacks by combining the power of word embedding and machine learning.

3.1 INTRODUCTION

Fraudsters engage in "Phishing Emails", a deceptive activity where they send emails that appear to be from trustworthy companies or organizations to deceive victims and

3. Phishing email detection framework using word embedding and machine learning

gain financial benefits through a camouflage email. They often include a link that looks legitimate but leads to a fake website where the victim is prompted to enter personal information. Any information the victim provides is directly sent to the scam artists behind the scheme.

Word embedding is a technique used to represent words in a numerical vector form. This vector representation can be used to capture the semantic meaning of words and phrases. Machine learning or deep learning algorithms are used to classify emails based on the vector representations of the header features.

Numerous studies (J Kuss et al. 2014; Kaltiala-Heino et al. 2004; Ryan et al. 2014; Shaw and Black 2008; Zhang et al. 2020) suggest that the younger generation has a strong dependence on digital devices and communication networks for various purposes, such as communication, education, entertainment, and financial transactions. As a result, this population is more vulnerable to phishing email attacks. Phishing attacks usually involve sending emails with links to forged websites that appear genuine. Email phishing is a significant and real threat to e-commerce since it involves messages from seemingly credible sources that request individuals and financial institutions to reveal and give access to sensitive information, which cybercriminals can then steal. According to the phishing activity trends report of the Anti-Phishing Working Group (APWG), the number of phishing attacks continued to increase during the fall of 2019. The APWG (2019b) report identified a total of 266,387 phishing sites from July through September 2019, which was 46% higher than the second quarter's 182,465 and nearly double the 138,328 recorded in APWG (2019a). However, email phishing activities drastically reduced in 2019, as shown in Table 3.1 of APWG's four-quarter statistics 2019 for unique phishing emails.

Table 3.1: APWG Email phishing statistics 2019

Quarter-1	Phishing Emails	Quarter-2	Phishing Emails	Quarter-3	Phishing Emails	Quarter-4	Phishing Emails
January	34630	April	37045	July	35530	October	45057
February	35364	May	40177	August	40457	November	42424
March	42399	June	34932	September	42273	December	45072
Total - Q1	112393	Total - Q2	112154	Total - Q3	118260	Total - Q4	132553

Table 3.1 provides data on the total number of unique phishing email messages received by the APWG from clients. The latest four-quarter reports in 2020 indicate that there were 1,031,347 unique phishing email campaigns recorded from consumers. Table 3.2 presents the monthly results for the four quarters of 2020. Cybercriminals have taken advantage of the COVID-19 pandemic by using disaster-related content to launch phishing attacks against healthcare facilities and professionals. However, the number of email phishing scams has decreased significantly in 2021, with only 484,469 cases reported.

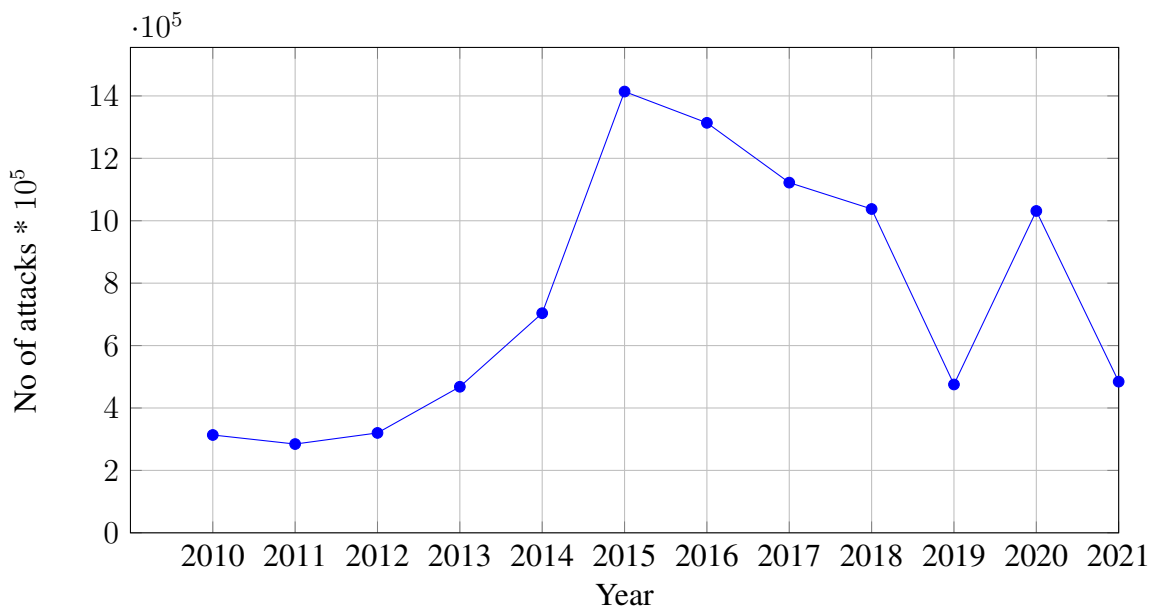


Figure 3.1: Email Phishing attacks from 2010 to 2021

Based on the APWG's 2015 survey report, the highest recorded number of phishing emails reached 1,413,978. The statistical data depicted in Figure 3.1 illustrates the trends of phishing email occurrences between 2010 and 2021. The data highlights that the year 2015 stands out as a critical period with a significant surge in phishing email attempts. The APWG survey emphasizes the need for additional research to develop effective anti-phishing solutions, as email phishing continues to pose a substantial threat.

Based on the Mimecast (2019) survey report, finance organizations, professional services, and manufacturing companies were the most affected by phishing attacks, with 68%, 66%, and 66%, respectively. Kaspersky (2019)'s third-quarter report highlights

3. Phishing email detection framework using word embedding and machine learning

that educational institutions and universities are at high risk of having their sensitive documents stolen and sold on the dark market. The current filters and security measures

Table 3.2: APWG Email phishing statistics 2020

Quarter-1	Phishing Emails	Quarter-2	Phishing Emails	Quarter-3	Phishing Emails	Quarter-4	Phishing Emails
January	52407	April	43282	July	119181	October	143950
February	43270	May	39908	August	119180	November	119700
March	44008	June	44497	September	128926	December	133038
Total - Q1	139685	Total - Q2	127687	Total - Q3	367287	Total - Q4	396688

have limited success in detecting phishing attacks, as phishing websites and emails are designed to resemble legitimate ones closely. This similarity makes it difficult for existing systems to distinguish between the two accurately, leaving users vulnerable to an increasing threat. Users become a victim and end up revealing their sensitive private information due to,

- Lack of computer system knowledge
- Inadequate knowledge of security and security indicators
- Replication of original sites with minor changes mostly goes unnoticed by users
- Ignoring security warnings.

3.2 WORD EMBEDDING:

Word embedding refers to a group of language models and methods for selecting features that are also known as word representation. Its main goal is to convert textual terms or phrases into a continuous low-dimensional space. Word embeddings effectively convert human textual language into a numeric representation. It is possible that the converted text to numbers is a different numeric representation of the same text.

3.2.1 Need of word embeddings:

Word embedding is a way of representing words as vectors of numbers in a high-dimensional space. The need for word embedding arises from natural language processing (NLP) algorithms often requiring a fixed-length representation of words to perform

various tasks such as language translation, sentiment analysis, text classification, and more. Word embedding techniques enable us to convert text data into a format machine learning models can understand and process. Instead of representing each word as a sparse one-hot vector, word embedding techniques transform each word into a dense vector that captures its semantic meaning and relationships with other words.

One of the significant advantages of using word embeddings is that they can capture the context and meaning of words, which is essential for many NLP applications. For example, words that are semantically similar or related in meaning, such as "cat" and "dog," will have vectors that are closer together in the embedding space than unrelated words like "cat" and "table." Word embeddings also allow us to apply mathematical operations such as vector addition and subtraction to find relationships between words. For instance, we can add the vector for "king" to the vector for "woman" and subtract the vector for "man" to get the vector that is closest to the vector for "queen." In summary, word embeddings play a critical role in many NLP applications by providing a way to represent words in a dense, low-dimensional space that captures their semantic relationships and meaning.

Example: Sentence = "Word embeddings are word converted into numbers". Words in the sentence are 'embeddings' or 'numbers'. A dictionary may be the list of all unique words in the sentence, so a dictionary may look like ['Word', 'embeddings', 'are', 'conver- ted', 'into', 'numbers']. Word embedding is categorized into two types: (1). Frequency-based and (2). Prediction-based embedding techniques.

3.2.2 Frequency-based word embedding

Frequency-based word embedding is a type of word embedding technique that uses the statistical information of a text corpus to generate word representations. The most common frequency-based word embedding technique is the Count-Based model, which includes methods like CountVectorizer and Term Frequency-Inverse Document Frequency (TF-IDF) vectorization.

Count Vectorization (CV): CountVectorizer is a technique that counts the frequency of each word in a text corpus and creates a vector of these counts for each

document. The resulting vectors can be used as features for a machine learning model. While this technique is simple and fast, it does not capture the semantic relationships between words.

Count vectorization involves generating a matrix with dimensions $d \times n$, where d represents the corpus size (the number of documents) and n corresponds to the number of distinct tokens found in the documents. This matrix records the frequency of each word's occurrence within a document. To measure the similarity between the resulting vectors, cosine similarity is employed, which evaluates the angle between the two vectors.

TF-IDF: TF-IDF vectorization is an extension of CountVectorizer that takes into account the relative importance of each word in the text corpus. It assigns a weight to each word in a document based on its frequency and frequency across all documents in the corpus. This weighting scheme aims to give more importance to words that are rare in the corpus but frequently occur in a specific document.

Term Frequency and Inverse Document Frequency vectorization works by finding out the unique words present not just in the document but in the entire corpus of the documents. The intuition behind this is that the more frequent words may not be the relevant words. Some words appear in the documents more number times. IDF works by finding such words and giving better unique words present in the entire corpus of documents, as the more frequent irrelevant words hold little to no new information. TF the number of times a word has appeared in a document. Further, it can be divided by the total number of words in a document. Therefore,

$$TF(t, d) = \frac{x}{y} \quad (3.1)$$

where x is the count of t in document d , and y is the number of words in document d .

IDF measures the uniqueness of the word across the corpus.

$$IDF(t, d) = \log\left(\frac{N}{n}\right) \quad (3.2)$$

where N is the total number of documents present in the corpus, and n is the number of documents where the term t appears.

$$TF - IDF = TF * IDF \quad (3.3)$$

CountVectorizer and TF-IDF vectorization generate sparse vectors, where most elements are zero, and each dimension corresponds to a particular word in the corpus vocabulary. While frequency-based word embedding techniques are simple and easy to implement, they have limitations. They do not capture the semantic relationships between phrases beyond their frequency, and they generate high-dimensional sparse vectors that can be computationally expensive to use in large datasets. Nonetheless, these methods are helpful as a starting point for many NLP tasks or as a complementary technique to more advanced word embedding models.

3.2.3 Prediction-based word embedding

Prediction-based word embeddings are more efficient and accurate language modeling techniques in modern research and are considered a byproduct of language models according to Almeida and Xexéo (2019). Some prediction-based word embedding techniques, such as Word2Vec, FastText, and GloVe, are discussed below.

Word2Vec (W2V): To infer any relationship between two words is difficult in their one-hot encoding representation. Sparsity is another issue with the one-hot encoding, as many redundant 0 are in their vector representation. Word2Vec solves these problems by using surrounding words to represent the target words. Word2Vec is a predictive model that learns embedding from the given text. It is a three-layer architecture with a small hidden layer that generates embeddings from given text. The size of the input and output provided is generally the same. Word2Vec has two algorithms: Continuous Bag of Words (CBOW) and SkipGram (SG).

- **CBOW:** Continuous Bag of Words tries to predict the word with the help of the context. This context can be a single word or a group of words. CBOW uses a neural network as continuous distributed representation of the context of words and predicts a word as an output. The working and architecture of the CBOW and SkipGram models were presented by Mikolov et al. (2013). This model predicts the probability of occurrence of a word given the context of words surrounding it.
- **SkipGram model:** The SkipGram model tries to predict the context of the given

3. Phishing email detection framework using word embedding and machine learning

word. The architecture is just opposite to that of the CBOW model. SkipGram takes the input as the target word and outputs the context words that surround the target word.

For example: Given the sentence, "It was an apple pie", if the input is "a", the output would be "It", "was", "apple", and "pie" for the window size of 5. The dimension of all the input and output data is the same, and one-hot encoded. This model consists of one hidden layer with a dimension equal to the embedding size, which is lesser than the vector size of input/output. A softmax activation function is applied at the end of the output layer, which describes the likelihood of the appearance of a specific word in the context.

Two challenges that appear with Word2Vec are,

- **Out of Vocabulary (OOV) words:** Word2Vec can handle only words it has encountered during its training. For example, Word2Vec vocabulary containing words such as "tensor" and "flow" can not handle embedding for the word "tensorflow", i.e., a compound word. Thus it leads to an "out of vocabulary" error.
- **Morphology:** Word2Vec does not do any parameter sharing for words such as "eat" and "eating" with the same radicals. Each word is uniquely learned based on the context in which the word appears. Thus the internal structure of the word can be utilized properly to make the embedding more efficient.

FastText: A library created by Facebook Artificial Intelligence Research (FAIR) laboratory, allows for learning word embedding and text classification (Bojanowski et al. 2017; Joulin et al. 2016a,b). It is open source software, that can learn billions of words in a few minutes and is suitable for supervised and unsupervised learning. This library implements the concept of enriching word vectors with subword information (Bojanowski et al. 2017) and a bag of tricks for efficient text classification (Joulin et al. 2016b). It is accessible in 294 languages, and can find resources on the Facebook research¹ page. The two methods used to learn word representation and develop word vectors in FastText are CBOW and SkipGram, as seen in Word2Vec.

¹<https://research.fb.com/downloads/fasttext/>

GloVe: Glove represents Global Vectors, an unsupervised learning distributed word representation model to obtain vector representation of words. Global Vectors generated using the co-occurrence matrix statistic from a corpus. The matrix denoted by X_i , $X_{i,j}$ represents the number of times the word j appears in the context of the word i . $P_{i,j} = X_{i,j}/X_i$, which gives the probability that the word j occurs in the context of the word i . These probabilities can provide some potential to encode some form of the underlying meaning of the contextual meaning.

3.3 MACHINE LEARNING

Machine learning is a type of artificial intelligence (AI) that allows computer systems to learn and improve their performance on a task without being explicitly programmed. It is a process of training algorithms to recognize patterns and make predictions or decisions based on data input. Machine learning algorithms categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning involves providing a machine learning model with labeled data, meaning the input data is paired with the correct output. The model then uses this labeled data to learn how to make predictions or classify new data.

On the other hand, unsupervised learning involves providing the model with unlabeled data and allowing it to identify patterns and relationships on its own. Clustering and dimensionality reduction are examples of unsupervised learning.

Reinforcement learning involves a model learning from feedback in the form of rewards or penalties, and using this feedback to improve its decision-making over time.

Machine learning has numerous applications in various fields, including image recognition, natural language processing, fraud detection, recommendation systems, and more. It is a rapidly growing field with the potential to revolutionize many industries and improve the accuracy and efficiency of many tasks.

3.4 EMAIL PHISHING

The invention of email is commonly credited to Ray Tomlinson, an American computer programmer, who implemented the first email system on the ARPANET (a precursor to

3. Phishing email detection framework using word embedding and machine learning

the internet) in 1971 (Leiner et al. 1997). Tomlinson is also credited with introducing the use of the ”@” symbol to separate the user from the destination address in an email address.

While Tomlinson is widely recognized as the inventor of email, other early email systems were in use at the time, such as the SNDMSG system on the Compatible Time-Sharing System (CTSS) developed by MIT in the 1960s. However, Tomlinson’s email system was the first to use the @ symbol and to establish the standard for sending and receiving messages across different computer networks. Email messages typically consist of three main components: the header, the body, and the footer. In this research, the primary focus is on analyzing the header and body components of the email. The taxonomy and structure of the email (Figures 3.2 & 3.3) clearly describe the format of the email message.

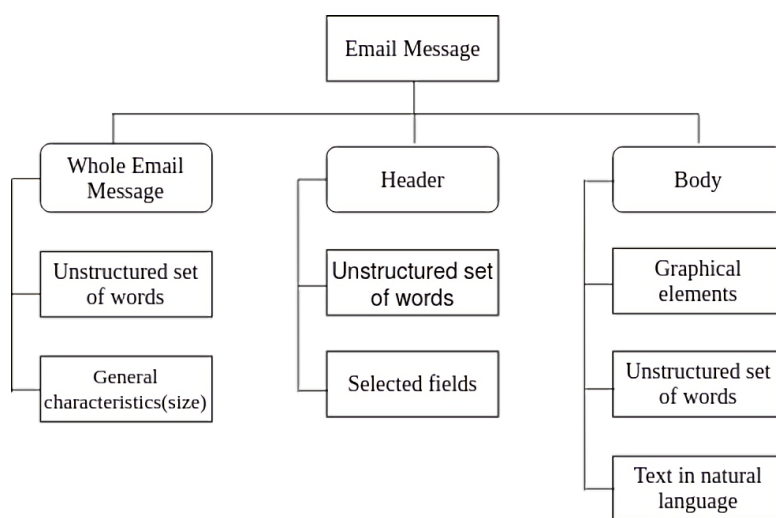


Figure 3.2: Email message Taxonomy - Courtesy (Almomani et al. 2013)

Header: The header contains information about the email, such as the sender and recipient(s), the date and time the message was sent, the subject line, and any other details such as the email’s importance level or whether it has been read or replied to.

Body: The email’s body is the message’s main content, where the sender writes the message to the recipient(s). It can include text, images, hyperlinks, attachments, and other multimedia elements. Fraudulent activities, such as phishing, can occur in either of the two components of an email. Email phishing is a cyber-attack where an attacker

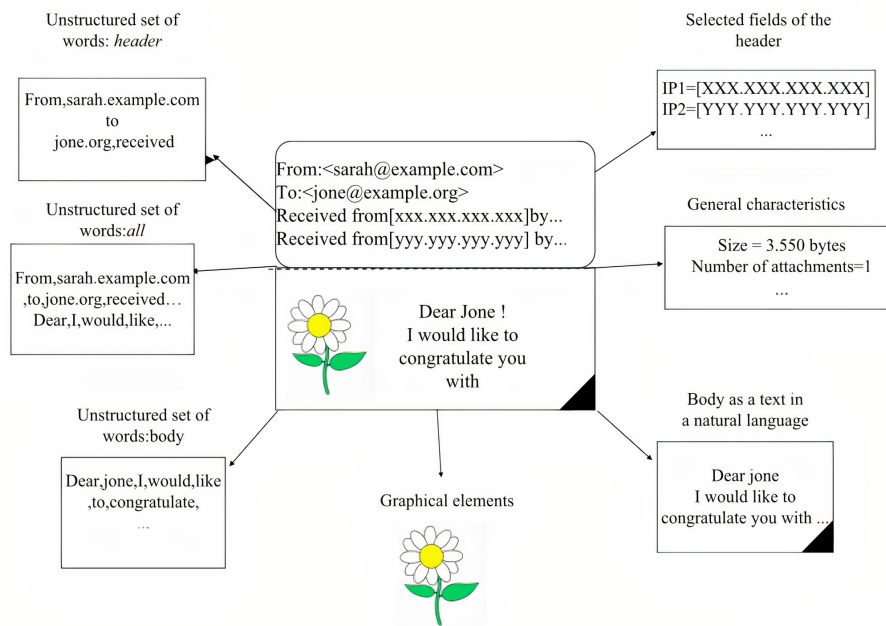


Figure 3.3: Email message structure - Courtesy (Almomani et al. 2013)

attempts to trick a victim into providing sensitive information such as login credentials, financial information, or personal information by sending fraudulent emails that appear to be from a trusted source. The attacker typically creates an email that looks like it came from a legitimate company or individual, then sends it to many recipients, hoping that some will fall for the scam.

The email usually contains a convincing message urging the recipient to take immediate action, such as clicking on a link, downloading an attachment, or entering their login credentials. The link or attachment may be malicious, and once clicked or downloaded, it can infect the victim's device with malware, allowing the attacker to gain access to their sensitive information. Alternatively, the login credentials entered by the victim may be captured by the attacker and used for malicious purposes.

To avoid falling for email phishing scams, it's essential always to be wary of unexpected or suspicious emails, particularly those that ask for sensitive information or urge immediate action. Always check the sender's email address, be wary of any discrepancies, and avoid clicking links or downloading attachments unless you know their authenticity. It's also a good idea to use anti-phishing software and keep your devices

3. Phishing email detection framework using word embedding and machine learning

and software up to date with the latest security patches.

Phishing attacks are categorized into deceptive and malware phishing. Deceptive phishing is related to social engineering schemes, which depend on forged email claims that appear as originated from a legitimate organization. And also, an embedded link redirects the user to fake websites to obtain personal information to defraud the user. The strategy of a phishing email is to attract victims and direct them to a particular phishing website. The received emails are embedded with URLs and trick the user into clicking on the embedded link to reveal confidential information. Many researchers have identified several email phishing features, and many different mechanisms exist to identify legitimate and fake emails. According to Almomani et al. (2013), three types of feature sets exist: basic features, latent topic features, and dynamic Markov chain features. The author (Almomani et al. 2013) identified different anti-phishing approaches contributed by researchers.

A brief survey of techniques used in machine learning, deep learning, word embedding, and natural language processing is discussed in detail in the previous chapter 2.

The *research contributions* of this chapter are listed below:

- Creation of In-house real-time phishing and legitimate email datasets.
- Proposed a novel phishing email detection technique using word embedding and machine learning.
- The proposed novel architecture uses only FOUR email header features and achieves competitive accuracy.
- Proposed work outperformed all other existing works on publicly available datasets.
- The proposed work justified that the newly created datasets are accurate and obtained nearly similar results as publicly available datasets.

A summary of major works based on machine learning and word embedding with machine learning or deep learning to detect phishing emails is discussed in Chapter 2. It

may be observed that the related works are also evaluated based on the parameters used. According to the survey, most of the research was carried out on both the header and body of an email, and some works on attachments along with the header and body of an email. None of the works focused exclusively on email header features or word vectors based on email headers. After analyzing the research gaps, this work proposes a model which uses only email header-based heuristics for efficient phishing email detection. The study of related works gave clarity to adopt word embedding for our research with machine learning classifiers. The architecture in the next section evaluates the best possible combination of the classification process with multiple word embedding and machine learning classifiers. Based on the research summary, it may be concluded that word embedding techniques may generate more suitable word vectors to classify given emails as phishing or legitimate using different machine learning classifiers. Hence, this chapter introduces a new approach for detecting phishing emails using machine learning and a novel word embedding technique.

3.5 PHISHING EMAIL DETECTION

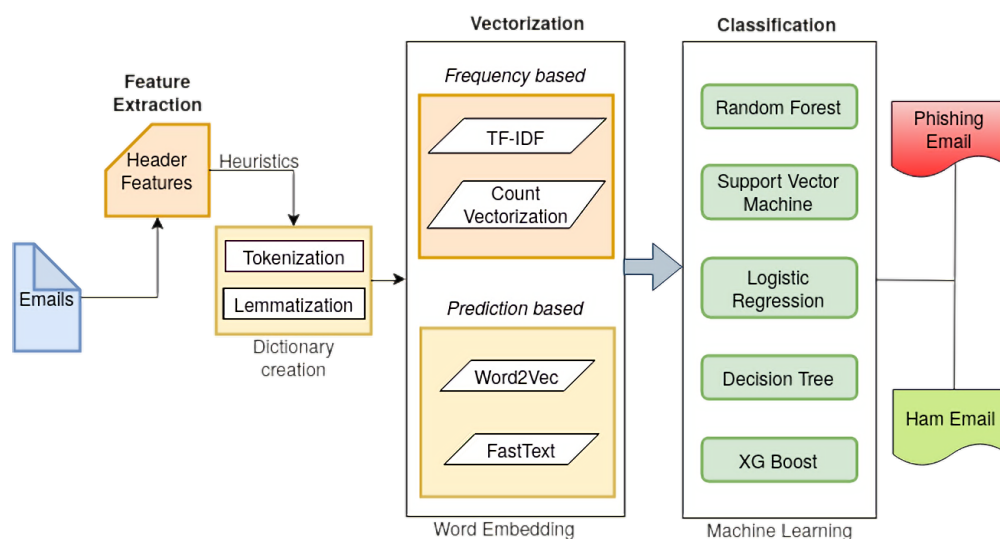


Figure 3.4: Architecture of Phishing Email Detection

Figure 3.4 illustrates the proposed design for identifying phishing emails, which comprises multiple stages to process and categorize emails as either legitimate or phishing. The process for classifying phishing emails involves five stages: 1. Input email

processing, 2. Feature extraction, 3. Dictionary creation, 4. Vectorization, and 5. Classification.

3.5.1 Input Emails

Email is the primary form of communication between known parties in electronic messaging media. The initial step in analyzing these emails is to organize them into dataset repositories. These repositories are then mined for the necessary heuristics. The required heuristics are obtained specifically from the email headers of pre-processed datasets sourced from public repositories and in-house generated ones.

3.5.2 Feature Extraction

The initial stage of the proposed model involves feature extraction, which entails obtaining the necessary heuristics from the emails. To accomplish this, Python scripts used to extract only the essential header heuristic features from MBOX format files. Following the extraction of the required heuristics, extraneous tags, text, garbage characters, and special symbols are eliminated. The extracted data from individual emails save to a CSV file. This CSV file serves as an input to the proposed architecture for distinguishing between legitimate and phishing emails.

3.5.3 Selected heuristic features

Chapter 2 acknowledged that previous research employed a mix of hybrid features or exclusively focused on content-based features. However, for this particular study, only four header labels have been selected as heuristic features. Specifically, the analysis will concentrate on the "From," "Return-Path," "Subject," and "Message-id" attributes found in the email header.

- **From:** The label "From" indicates the sender's name and email address, which may belong to an individual, a company, or an organization that has registered an account with an email service provider. Unfortunately, fraudsters may also use genuine email accounts to send harmful content, such as web links, malware, and other deceptive tactics to scam unsuspecting users.
- **Return-Path:** The "Return-path" is an attribute within the email header created

by the SMTP (Simple Mail Transfer Protocol) protocol. Its primary function is to keep a record of the reverse path of an email, mainly used for managing bounced emails. However, it is worth noting that cybercriminals can exploit these bounced email return paths to distribute harmful links and try to deceive specific individuals into revealing their sensitive information.

- **Subject:** The subject of an email provides a concise overview of the message content, and it is a crucial heuristic among the four header fields. Cybercriminals often exploit the subject line to include attention-grabbing, time-sensitive language and terms related to banks, finances, and accounts in their attempts to carry out fraudulent activities.
- **Message-ID:** A Message-ID is a distinctive identifier assigned to each email that adheres to a particular format. Since the Message-ID is specific to the email address and message, no two emails will share the same ID. Regrettably, cybercriminals can use a generated Message-ID to pose as legitimate users and engage with the end-user, potentially obtaining their personal information.

The above four email header features are useful in classifying phishing emails for the following reasons:

The "From" field indicates the sender of the email. Phishing emails often impersonate legitimate entities or individuals to deceive recipients. By analyzing the "From" field, suspicious or spoofed email addresses can be identified, helping to flag potential phishing attempts.

The "Return-path" header provides information about the email's path and assists in managing bounced emails. In phishing attacks, cybercriminals may manipulate the return-path to hide their true identity or mask the actual source of the email. Analyzing this header can reveal discrepancies or anomalies that may indicate a phishing attempt.

The "Subject" line of an email often serves as the initial attention-grabbing element. Phishing emails may employ enticing or urgent subjects to prompt recipients to take immediate action. By examining the subject line, suspicious keywords, patterns, or

3. Phishing email detection framework using word embedding and machine learning

unusual requests can be identified, helping to identify potential phishing emails.

The "Message-ID" is a unique identifier assigned to each email. While this header alone may not directly contribute to phishing email classification, it can aid in tracking and analyzing email conversations, identifying patterns, and detecting campaigns or patterns associated with phishing attacks.

By analyzing and scrutinizing these specific email header features, security systems and algorithms can leverage their information to assess the authenticity and potential malicious intent of emails, thereby assisting in the classification and identification of phishing emails.

3.5.4 Dictionary creation

The heuristics extracted from the text undergo tokenization using the *nlTK* library, which breaks down the input document into smaller units like words or terms. Depending on the input document, it could be a phrase, sentence, paragraph, or an entire document, and each of these smaller units is called a token. The tokenization output is then used for lemmatization, where the inflectional endings of words are removed, and the dictionary form of the word is provided through the use of vocabulary and morphological analysis that studies the structure and formation of the word. Lemmatization accurately identifies a word and converts it to its base form, considering the context in which it is used.

Example: Let us consider the following sample feature,

Subject = "Transaction alerts for your State Bank of India Debit Card"

When the subject is tokenized, the string looks like,

```
['Transaction', 'alerts', 'for', 'your', 'State', 'Bank', 'of', 'India', 'Debit', 'Card']
```

The output from the tokenizer is fed to the lemmatizer, the obtained output from the lemmatizer as below,

```
['Transaction', 'alert', 'for', 'your', 'State', 'Bank', 'of', 'India', 'Debit', 'Card']
```

3.5.5 Vectorization

The vectorization process, an unsupervised learning method, involves converting words to a numerical format subsequently trained using classification models. Before vectorization, pre-processing extracted features from the emails involves removing special symbols, extra spaces, irrelevant numeric data, and garbage characters as part of the tokenization process. The extracted words are lemmatized to remove inflectional endings and converted to lowercase. This work employs two common vectorization methods: frequency-based (count of words/context co-occurrences) and prediction-based. The prediction-based methods include FastText and Word2Vec, while the frequency-based methods include TF-IDF and Count vectorization. These word embedding techniques allow for the representation of words, enabling machine learning algorithms to comprehend words with comparable meanings. Word vectors are numerical vectors representing words meaning and are multidimensional floating-point values that approximate positions in geographic space for semantically similar words.

The Word2Vec SkipGram algorithm takes a dictionary of existing words as input and generates corresponding vectors for each word, even for unfamiliar ones. The algorithm incorporates numerous parameters such as vector size, window size, minimum count, number of workers, number of iterations, and input datasets. By adjusting these parameters, appropriate vectors can be generated to optimize performance.

The Word2Vec-CBOW model utilizes the same parameters as SkipGram to generate word vectors comprised of real numbers, contingent on the vector size and other parameters designated to the Word2Vec function. Compared to SkipGram, Word2Vec's CBOW technique predicts the target word by taking context words as input. This technique is quicker and more effective with more common words.

The FastText library function is equipped with various parameters, such as the input corpus, vector size, window size, minimum count, and number of workers, used to generate vectors. The algorithm employs hierarchical classifiers and n-gram techniques to train models with unlabeled datasets. In this study, the FastText SkipGram and CBOW models take a dictionary of words generated in Section 3.5.4 as input and

produce corresponding real-valued vectors.

TF-IDF: TF-IDF is a technique that helps to identify the most important words in a document by computing a weight for each word that is proportional to its frequency in the document but inversely proportional to its frequency in the entire corpus of documents. The TF part of the technique calculates the frequency of a word in a document, while the IDF part calculates the importance of a word in the corpus of documents. The resulting vectors are generated by replacing the true or false conditioned boolean values of vector size for the words of the dictionary. TF-IDF terms have to be normalized to reduce the bias in term frequency from terms in short or longer documents. The corresponding sparse matrix is generated to identify the corpus's term frequency and inverse document frequency.

Count Vectorization: The Count Vectorizer function performs basic pre-processing on the input text, which includes removing punctuation marks and converting all words to lowercase. It then creates a vocabulary of known words that will be used to encode unseen text later. The encoded vector returned has a length equal to the size of the entire vocabulary, and the integer count indicates the number of times each word appeared in the document. To reduce bias in term frequency from terms in short or long documents, normalization of Count Vectorizer terms is necessary. The normalized resultant vector is size 100, and the corresponding sparse `cse_matrix` is obtained from the count vectorizer. The generated vectors are further used in machine learning classifiers to classify the email as phishing or legitimate. Section 3.2 discusses the workings of the word embedding algorithms used in the proposed work.

3.5.6 Classification

The vectorization module actively generates a variety of vectors that excel at determining word similarity within a continuous vector space. We use the vectors produced by word embedding techniques in conjunction with five machine-learning classification algorithms to classify emails as phishing or legitimate. The work demonstrates the use of machine learning algorithms such as Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), XGBoost, and Logistic Regression (LR) for email classifica-

tion. These algorithms are well known for their ability to correctly classify emails as phishing or legitimate.

3.6 IMPLEMENTATION

The presented model relies on several libraries for its implementation. The Pandas library used for efficient database processing and data manipulation, including reshaping and merging data frames. The nltk library, on the other hand, is used for performing statistical natural language processing tasks such as tokenization, lemmatization, and stopword removal. In addition, the sklearn library is used for various machine learning tasks, including classification, regression, clustering, and dimensionality reduction. This library provides several algorithms and tools to construct robust machine learning models. Lastly, the Gensim library is an open-source tool for creating word embeddings. It provides an efficient way to convert words and documents into vectors that can be used in various natural language processing tasks, including text similarity and document classification.

Overall, combining these libraries provides a powerful toolset for developing the model. They enable efficient data processing, feature engineering, and the creation of robust machine learning models, which are essential for accurate phishing email detection.

3.7 RESULTS AND DISCUSSION

The experiments are conducted using three different datasets, as mentioned in Table 2.8. The training and testing were performed on each dataset using 70% & 30% of their total size and a window size 10. However, before performing the main experiments, some prerequisite tasks were required. Developed Python scripts to minimize the noise by removing empty spacing, angle brackets, single and double quotes, and other unnecessary symbols. The parsed email header heuristics were saved as a CSV file. The proposed model trained using these selected CSV files with word embedding and ML algorithms. The training accuracy from dataset-1 was 100% with Word2Vec (CBOW & SkipGram) and RF for vector size 300. Similarly, the highest training ac-

3. Phishing email detection framework using word embedding and machine learning

curacy achieved in the training experiments conducted on dataset-2 was 99.88% with Word2Vec (CBOW) and RF for vector size 200. The training was also conducted on a purely in-house repository, i.e., dataset-3, and the highest training accuracy achieved was 99.87% with Word2Vec (CBOW & SkipGram) and RF for vector size 150. The results indicate that Word2Vec with RF consistently performed the best with all three datasets.

3.7.1 Experiment-1

The model presented in this study utilized Dataset-1 as input and employed various word embedding algorithms, such as TF-IDF, Count Vectorization, Word2Vec (CBOW), Word2Vec (SkipGram), FastText (CBOW), and FastText (SkipGram), to conduct the experiment. To assess the performance of each algorithm, the output vector size was varied from 50 to 300, and the corresponding results were recorded in Table 3.3. Notably, when the vector size was set to 300, TF-IDF in combination with RF, DT, XG Boost, and LR algorithms demonstrated high accuracy levels of 99.37%, 99.13%, 98.62%, and 99.07% respectively. Similarly, for vector size 150, TF-IDF in combination with SVM achieved an accuracy of 98.16%. Count Vectorizer exhibited similar performance to TF-IDF. The RF, DT, XG Boost, LR, and SVM algorithms achieved accuracies of 99.33%, 98.92%, 98.42%, 98.77%, and 97.17% respectively for both vector sizes of 300 and 150. The results indicate that, in most cases, the efficiency of TF-IDF and Count Vectorization improves with increasing vector size, except for SVM.

The performance of Word2Vec (CBOW) was evaluated alongside RF, DT, and SVM algorithms, achieving accuracy levels of 99.26% and 98.79% for vector size 300, and 98.90% for vector sizes 100 and 200, respectively. When combined with XG Boost and LR algorithms, Word2Vec (CBOW) achieved an accuracy of 98.92% for vector size 100 and 99.16% for vector size 200. Word2Vec (SkipGram) in combination with RF, DT, and LR algorithms yielded accuracies of 99.35%, 98.96%, and 99.26%, respectively, for vector size 200. Similarly, SVM and XG Boost algorithms achieved accuracies of 98.90% and 98.92%, respectively, for vector sizes 50 and 150. Notably, the results of Word2Vec exhibit variability across the chosen vector sizes.

Table 3.3: Selection of vector size with Dataset-1

Word Embedding	Machine Learning	Testing Accuracy(%) of Vectors				
		50	100	150	200	300
TF-IDF	RF	98.92	98.59	99.07	99.16	99.37
	DT	98.49	98.03	98.64	98.66	99.13
	SVM	97.08	97.73	98.16	98.08	97.97
	XG Boost	98.25	97.64	98.27	98.27	98.62
	LR	98.21	98.16	98.62	98.92	99.07
Count Vectorizer	RF	98.90	98.59	98.98	99.29	99.33
	DT	98.55	98.08	98.55	98.87	98.92
	SVM	96.11	97.08	97.17	97.10	96.37
	XG Bosst	98.33	97.86	98.34	98.38	98.42
	LR	98.18	97.77	98.53	98.64	98.77
Word2Vec (CBOW)	RF	98.81	98.94	98.87	99.09	99.26
	DT	97.73	98.25	98.49	98.40	98.79
	SVM	98.72	98.90	98.85	98.90	98.75
	XG Boost	98.70	98.92	98.40	98.79	98.79
	LR	99.15	98.90	98.92	99.16	98.90
Word2Vec (SkipGram)	RF	98.94	99.00	98.98	99.35	98.79
	DT	98.29	98.23	98.44	98.96	98.68
	SVM	98.90	98.75	98.77	98.85	98.46
	XG Boost	98.75	98.57	98.92	98.77	98.66
	LR	99.11	99.07	99.16	99.26	98.92
FastText (CBOW)	RF	99.42	99.50	99.50	99.31	99.16
	DT	98.92	99.03	99.11	98.94	98.64
	SVM	98.38	98.29	97.95	97.79	97.45
	XG Boost	98.87	99.29	98.64	98.94	98.70
	LR	98.66	99.05	98.72	98.94	98.55
FastText (SkipGram)	RF	99.44	99.33	99.39	99.37	99.46
	DT	98.75	98.94	99.24	98.94	98.79
	SVM	98.87	98.46	98.31	97.86	97.79
	XG Boost	99.26	98.96	99.18	98.85	98.96
	LR	99.44	99.16	99.03	99.37	99.24

FastText, a vectorization algorithm introduced by Facebook, is similar to Word2Vec which employs two-word learning techniques, CBOW and SkipGram. FastText (CBOW) combined with RF achieves the highest accuracy of 99.50% for vector sizes 100 and 150, according to the results. Similarly, for vector sizes 150 and 50, the DT and SVM algorithms achieve accuracies of 99.11% and 98.38%, respectively. For vector size 100, the XG Boost and LR algorithms achieve accuracies of 99.29% and 99.05%, respectively. When combined with RF, FastText (SkipGram) achieves an accuracy of

3. Phishing email detection framework using word embedding and machine learning

99.46% for vector size 300. Furthermore, when applied to DT, FastText (SkipGram) achieves 99.24% accuracy for vector size 150, while SVM, XG Boost, and LR algorithms achieve 99.87%, 99.26%, and 99.44% accuracy for vector size 50, respectively. FastText (CBOW) combined with RF achieves the highest level of accuracy of 99.50% for vector size 100.

Table 3.4: Selection of vector size with Dataset-2

Word Embedding	Machine Learning	Accuracy(%) of Vector size				
		50	100	150	200	300
TF-IDF	RF	95.99	98.47	99.05	99.39	99.28
	DT	95.51	97.86	98.42	98.77	98.50
	SVM	94.96	97.20	97.80	97.94	97.74
	XG Boost	95.44	97.91	98.36	98.66	98.67
	LR	95.00	97.38	98.34	98.83	98.94
Count Vectorizer	RF	95.96	98.51	99.01	99.37	99.37
	DT	95.78	98.02	98.40	98.62	98.56
	SVM	94.93	96.69	97.33	97.53	96.68
	XG Boost	95.42	98.0	98.40	98.70	98.77
	LR	95.10	97.14	98.28	98.73	98.75
Word2Vec (CBOW)	RF	98.39	98.58	98.47	98.42	98.62
	DT	97.44	97.43	97.88	97.70	97.60
	SVM	97.46	98.60	98.40	98.33	98.40
	XG Boost	98.03	97.88	98.12	98.10	98.09
	LR	97.91	98.16	98.21	98.13	98.38
Word2Vec (SkipGram)	RF	98.68	98.58	98.72	98.38	98.56
	DT	97.78	97.50	97.44	97.88	97.86
	SVM	97.97	97.99	97.87	97.35	97.53
	XG Boost	98.23	98.06	98.02	97.84	98.11
	LR	98.61	98.77	98.68	98.70	98.96
FastText (CBOW)	RF	98.72	98.62	98.58	98.39	98.64
	DT	97.94	98.00	97.98	97.44	97.64
	SVM	97.98	97.61	97.35	96.82	96.78
	XG Boost	98.15	98.08	97.85	97.61	97.83
	LR	98.00	97.85	97.87	97.30	97.46
FastText (SkipGram)	RF	98.48	98.47	98.71	98.59	98.50
	DT	97.54	97.75	97.48	97.77	97.49
	SVM	97.24	97.16	97.05	96.82	96.65
	XG Boost	98.05	98.03	97.93	98.11	98.10
	LR	98.41	98.36	98.43	98.50	98.30

3.7.2 Experiment-2

Table 3.4 shows the results of Experiment-1 on Dataset-2, which aimed to validate our proposed technique using a legitimate dataset generated in-house. For a vector size of 200, the accuracy results for TF-IDF with RF, DT, and SVM are 99.39%, 98.77%, and 97.94%, respectively. XG Boost and LR achieved 98.67% and 98.94% accuracy for vector size 300, respectively. Count vectorizer also performed well, with RF, DT, and SVM achieving accuracies of 99.37%, 98.62%, and 97.53% for vector size 200, respectively. XG Boost and LR also achieved accuracies of 98.77% and 98.75%, respectively. It is worth noting that the TF-IDF and count vectorizer performed better for vector sizes 200 and 300.

The accuracy achieved with Word2Vec (CBOW) combined with RF for vector size 300 was 98.62%. For vector sizes 150, 100, 150, and 300, DT, SVM, XG Boost, and LR achieved accuracies of 97.88%, 98.60%, 98.12%, and 98.38%, respectively. With Word2Vec (SkipGram) and RF, an accuracy of 98.72% was obtained for vector size 150. Similarly, DT, SVM, XG Boost, and LR achieved accuracies of 97.88%, 97.99%, 98.23%, and 98.96% for vector sizes 200, 100, 50, and 300, respectively.

For vector size 50, CBOW with RF, SVM, XG Boost, and LR achieved accuracies of 98.72%, 97.98%, 98.15%, and 98.00%, respectively. For vector size 100, DT achieved an accuracy of 98.00%. For vector size 150, FastText (SkipGram) combined with RF achieved an accuracy of 98.71%. FastText (SkipGram) achieved accuracies of 97.77%, 97.24%, 98.11%, and 98.50% when combined with DT, SVM, XG Boost, and LR for vector sizes 200, 50, 200, and 200, respectively.

Overall, the results of the experiments performed on Dataset-2 show that the accuracy varies with vector size. The highest accuracy achieved with TF-IDF and RF for vector size 200 was 99.39%.

3.7.3 Experiment-3

In the current experiment, Dataset-3, an internal repository, serves as the input for the proposed model. The results of this experiment are presented in Table 3.5. The following results were obtained by using Dataset-3 as the input for the proposed model: For

3. Phishing email detection framework using word embedding and machine learning

vector sizes 150 and 200, TF-IDF combined with RF achieved an accuracy of 99.12%, while TF-IDF combined with DT achieved an accuracy of 99.03%. SVM, XG Boost, and LR accuracy rates for vector sizes 100 and 150 were 97.60%, 98.74%, and 98.34%, respectively. When combined with RF, DT, SVM, XG Boost, and LR, Count Vectorizer produced accuracy rates of 99.18%, 99.09%, 97.63%, 98.86%, and 98.49% for vector sizes 150, 200, 150, 200, and 150, respectively. For a vector size of 150, RF and SVM achieved an accuracy of 98.86% in Word2Vec (CBOW). DT, XG Boost, and LR achieved accuracy rates of 98.69%, 98.52%, and 98.54% for vector sizes 100, 50, and 200, respectively. In the case of Word2Vec (SkipGram), RF achieved 99.06% accuracy for a vector size of 100, while DT, XG Boost, and LR achieved 98.69%, 98.76%, and 98.86% accuracy for a vector size of 300, respectively. Similarly, for a vector size of 50, SVM achieved an accuracy of 98.69%. Overall, the results of the experiment with Dataset-3 show that the accuracy rates achieved for different vector sizes vary across the algorithms used.

FastText (CBOW) achieved 98.86% accuracy with RF for a vector size of 150. With a vector size of 200, the accuracy rates of DT, XG Boost, and LR were 98.76%, 98.62%, and 98.46%, respectively. With a vector size of 50, SVM achieved an accuracy rate of 98.16%. FastText (SkipGram) combined with RF, SVM, and XG Boost, on the other hand, achieved accuracy rates of 98.85%, 98.11%, and 98.52% for a vector size of 50, respectively. For a vector size of 300, DT and LR achieved accuracy rates of 98.62% and 98.67%, respectively. Overall, for a vector size of 150, the Count Vectorizer combined with RF produced the highest accuracy rate of 99.18%.

3.7.4 Performance Evaluation with Dataset-1

In the current study, the presented model is assessed using individual datasets, and the results obtained from Dataset-1 are presented in Table 3.6, which includes six metrics. After analyzing the results in Section 3.7.1 and Table 3.3, a vector size of 300 was chosen. Based on the input from Dataset-1, the RF classifier with FastText (SkipGram) exhibited the highest accuracy of 99.46%, along with MCC, Precision, TPR, TNR, and F-Score metrics of 98.99%, 99.89%, 99.20%, 99.84%, and 99.54%, respectively. The

Table 3.5: Selection of vector size with Dataset-3

Word Embedding	Machine Learning	Accuracy(%) of Vector size				
		50	100	150	200	300
TF-IDF	RF	98.73	99.06	99.12	99.12	99.08
	DT	98.58	98.71	98.70	99.03	98.93
	SVM	96.46	97.60	97.60	97.05	97.03
	XG Boost	98.24	98.74	98.52	98.69	98.64
	LR	97.64	98.00	98.34	98.16	98.08
Count Vectorizer	RF	98.75	99.06	99.18	99.17	99.12
	DT	98.67	98.92	98.79	99.09	98.93
	SVM	96.32	97.49	97.63	97.04	96.93
	XG Boost	98.49	98.72	98.73	98.86	98.79
	LR	97.71	98.08	98.49	98.09	98.09
Word2Vec (CBOW)	RF	98.75	98.80	98.86	98.74	98.80
	DT	98.57	98.69	98.60	98.52	98.57
	SVM	96.83	98.09	98.50	98.22	98.26
	XG Boost	98.52	98.46	98.47	98.02	98.08
	LR	98.18	97.94	98.24	98.54	98.19
Word2Vec (SkipGram)	RF	98.81	99.06	99.02	98.63	98.99
	DT	98.30	98.58	98.50	98.43	98.69
	SVM	98.69	98.31	98.27	98.03	98.12
	XG Boost	98.04	98.38	98.59	98.46	98.76
	LR	98.38	98.47	98.78	98.78	98.86
FastText (CBOW)	RF	98.79	98.82	98.86	98.75	98.80
	DT	98.50	98.63	98.64	98.76	98.68
	SVM	98.16	97.98	97.75	97.88	97.89
	XG Boost	98.26	98.46	98.26	98.62	98.32
	LR	98.11	98.35	98.31	98.46	98.27
FastText (SkipGram)	RF	98.85	98.60	98.84	98.73	98.84
	DT	98.49	98.49	98.53	98.51	98.62
	SVM	98.11	97.42	97.48	97.16	97.22
	XG Boost	98.52	98.37	98.46	98.41	98.26
	LR	98.46	98.42	98.47	98.47	98.67

table also highlights the top individual classifier accuracies, with SVM using Word2Vec (CBOW) achieving an accuracy of 98.75%. Additionally, XG Boost and LR with FastText (SkipGram) achieved their best individual accuracies of 98.96% and 99.24%, respectively.

3. Phishing email detection framework using word embedding and machine learning

Table 3.6: Performance Evaluation with Dataset-1

Word Embeddings	Algorithm	Acc	MCC	Precision	TPR	TNR	F-Score
TF-IDF	RF	99.37	98.71	99.96	98.98	99.95	99.47
	DT	99.13	98.22	99.48	99.05	99.26	99.26
	SVM	97.97	95.84	99.63	96.99	99.45	98.30
	XGBoost	98.62	97.15	99.30	98.36	98.99	98.83
	LR	99.07	98.09	99.70	98.72	99.57	99.21
Count Vectorizer	RF	99.33	98.62	99.85	99.01	99.79	99.43
	DT	98.92	97.77	99.26	98.90	98.95	99.08
	SVM	96.37	92.58	99.08	94.95	98.61	96.97
	XGBoost	98.42	96.75	99.22	98.11	98.88	98.66
	LR	98.77	97.46	99.37	98.54	99.10	98.95
Word2Vec (CBOW)	RF	99.26	98.50	100	98.75	100	99.37
	DT	98.79	97.52	98.95	98.95	98.56	98.95
	SVM	98.75	97.44	99.89	97.98	99.84	98.93
	XGBoost	98.79	97.53	99.77	98.16	99.68	98.96
	LR	98.90	97.74	99.40	98.70	99.17	99.05
Word2Vec (SkipGram)	RF	98.79	97.53	100	97.95	100	98.96
	DT	98.68	97.30	99.29	98.44	99.01	98.86
	SVM	98.46	96.88	99.92	97.49	99.89	98.69
	XGBoost	98.66	97.27	99.89	97.84	99.84	98.85
	LR	98.92	97.79	99.85	98.31	99.79	99.07
FastText (CBOW)	RF	99.16	98.26	99.82	98.77	99.73	99.29
	DT	98.64	97.18	99.05	98.65	98.61	98.85
	SVM	97.45	94.72	98.39	97.33	97.63	97.86
	XGBoost	98.70	97.32	99.60	98.24	99.40	98.91
	LR	98.55	97.00	98.98	98.58	98.51	98.78
FastText (SkipGram)	RF	99.46	98.88	99.89	99.20	99.84	99.54
	DT	98.79	97.50	98.76	99.19	98.21	98.97
	SVM	97.79	95.48	99.67	96.70	99.50	98.15
	XGBoost	98.96	97.85	99.56	98.69	99.36	99.12
	LR	99.24	98.44	99.71	99.02	99.57	99.36

3.7.5 Performance Evaluation with Dataset-2

The testing procedures for dataset-2 are identical to those discussed for dataset-1 in the previous section 3.7.4. A vector size of 200 is selected based on the results obtained in experiment-2, as indicated in Table 3.4. Table 3.7 presents the testing results of dataset-2, with seven different metrics. Among all the results, TF-IDF demonstrated excellent performance with RF, DT, and LR, achieving accuracies of 99.39%, 98.77%, and 98.83%, respectively. SVM and XG Boost showed good results with FastText

(SkipGram) and Count vectorizer, with accuracies of 98.82% and 98.70%, respectively, as highlighted in Table 3.7. The highest accuracy achieved is observed for TF-IDF with RF, with corresponding metrics of MCC 98.63%, Precision 99.19%, TPR 98.97%, TNR 99.60%, and F-score of 99.08%.

Table 3.7: Performance Evaluation with Dataset-2

Word Embeddings	Algorithm	Acc	MCC	Precision	TPR	TNR	F-Score
TF-IDF	RF	99.39	98.63	99.19	98.97	99.60	99.08
	DT	98.77	97.23	98.16	98.13	99.09	98.14
	SVM	97.94	95.35	96.55	97.22	98.29	96.88
	XGBoost	98.66	96.98	98.31	97.66	99.16	97.98
	LR	98.83	97.36	98.24	98.24	99.13	98.24
Count Vectorizer	RF	99.37	98.57	99.30	98.79	99.65	99.05
	DT	98.62	96.89	97.76	98.08	98.89	97.92
	SVM	97.93	94.41	95.37	97.12	97.72	96.24
	XGBoost	98.70	97.07	98.38	97.70	99.20	98.04
	LR	98.73	97.14	98.09	98.09	99.05	98.09
Word2Vec (CBOW)	RF	98.42	96.50	99.78	95.60	99.89	97.64
	DT	97.70	94.80	97.07	95.97	98.56	96.52
	SVM	98.33	96.27	98.96	96.08	99.48	97.50
	XGBoost	98.10	95.78	99.15	95.26	99.57	97.17
	LR	98.13	95.76	97.44	96.87	98.75	97.15
Word2Vec (SkipGram)	RF	98.38	96.43	99.52	95.77	99.76	97.61
	DT	97.88	95.25	97.43	96.24	98.72	96.83
	SVM	97.35	94.21	99.12	93.31	99.55	96.13
	XGBoost	97.85	95.22	98.42	95.25	99.20	96.81
	LR	98.70	97.08	98.57	97.53	99.28	98.05
FastText (CBOW)	RF	98.39	96.48	99.18	96.23	99.57	97.68
	DT	97.44	94.33	96.79	95.76	98.33	96.27
	SVM	96.82	93.01	96.79	94.07	98.31	95.41
	XGBoost	97.61	94.77	98.25	94.93	99.08	96.56
	LR	97.30	94.04	97.29	94.92	98.58	96.09
FastText (SkipGram)	RF	98.59	96.87	99.56	96.31	99.78	97.90
	DT	97.77	94.98	96.77	96.52	98.40	96.64
	SVM	98.82	93.09	98.82	92.16	99.39	95.38
	XGBoost	98.11	95.82	99.01	95.47	99.50	97.21
	LR	98.50	96.63	98.05	97.45	99.03	97.75

3.7.6 Performance Evaluation with Dataset-3

The testing procedures for dataset-3 were conducted in a similar way to those discussed in sections 3.7.1 and 3.7.2, and the results are presented in Table 3.9. These results are comparable to those obtained for dataset-1 and dataset-2, demonstrating that the in-house dataset creation procedures are reliable. Our institution's real-time repository, which is unique to us, is a noteworthy contribution. The best accuracy achieved with the new corpus was 99.18% using Count Vectorizer and RF, with corresponding MCC, Precision, TPR, TNR, and F-score of 98.11%, 98.05%, 99.38%, 99.09%, and 98.71% respectively.

3.7.7 Model Validation

The phishing email classification uses a validation method of a train/test split with a 70%/30% ratio of the total dataset size for all three datasets. This approach randomly divides the data into 70% for training and 30% for testing. The confusion matrices for the results of each individual dataset can be found in Tables 3.8 - (a), (b), and (c). Dataset-1 contains 15430 emails, with 10801 used for training and 4629 for testing. Dataset-2 has 27405 emails, with 19183 for training and 8222 for testing. Dataset-3 has a total of 27256 emails, with 19079 used for training and 8177 for testing. The evaluation metrics computed from the confusion matrix for the binary classifier are tabulated in Table 3.10.

It is worth noting that the training time for various algorithms varies from 67.15 seconds (for TF-IDF) to 425.02 seconds (for Word2Vec-SkipGram) when the vector size is set to 200. Similarly, the testing time for various algorithms ranges from 50.44 seconds (for TF-IDF) to 328.56 seconds (for Word2Vec-SkipGram) when the vector size is set to 200.

3.7.8 Performance of individual features

The current study employs four header labels exclusively for email classification. The performance of each label is presented in Table 3.11. Notably, the Return-Path feature obtains a high accuracy of 99.34% when used in conjunction with FastText and RF. Similarly, Subject is the second-best feature, achieving an accuracy of 98.71% when

Table 3.8: Confusion Matrix (a). Dataset-1, (b). Dataset-2, (c). Dataset-3

	True Positive	True Negative		TP	TN		TP	TN
False Positive	2741	1	FP	2701	22	FP	2652	54
False Negative	22	1865	FN	28	5471	FN	13	5458
	(a)			(b)			(c)	

Table 3.9: Performance Evaluation with Dataset-3

Word Embeddings	Algorithm	Acc	MCC	Precision	TPR	TNR	F-Score
TF-IDF	RF	99.12	97.97	97.86	99.38	99.00	98.61
	DT	98.70	97.03	98.28	97.68	99.19	97.98
	SVM	97.60	94.52	92.62	99.87	96.64	96.11
	XGBoost	98.52	96.59	97.17	98.18	98.67	97.67
	LR	98.33	96.17	95.76	99.01	98.03	97.36
Count Vectorizer	RF	99.18	98.11	98.05	99.38	99.09	98.71
	DT	98.79	97.22	98.20	98.01	99.15	98.11
	SVM	97.63	94.58	92.59	100	96.63	96.15
	XGBoost	98.73	97.07	97.63	98.38	98.89	98.00
	LR	98.49	96.55	95.64	99.64	97.99	97.60
Word2Vec (CBOW)	RF	98.86	97.39	98.51	97.95	99.29	98.23
	DT	98.60	96.82	98.59	97.11	99.33	97.84
	SVM	98.51	96.57	96.53	98.79	98.38	97.65
	XGBoost	98.47	96.49	97.49	97.75	98.81	97.62
	LR	98.24	95.98	97.90	96.65	99.00	97.28
Word2Vec (SkipGram)	RF	99.02	97.78	98.51	98.51	99.27	98.51
	DT	98.49	96.61	98.48	97.00	99.24	97.73
	SVM	98.27	96.12	94.84	99.92	97.52	97.31
	XGBoost	98.59	96.82	98.03	97.71	99.03	97.87
	LR	98.78	97.24	98.40	97.89	99.21	98.15
FastText (CBOW)	RF	58.86	97.40	96.97	99.49	98.57	98.21
	DT	98.64	96.89	97.54	98.24	98.83	97.89
	SVM	97.75	94.88	93.10	99.92	96.82	96.39
	XGBoost	98.26	96.02	96.78	97.82	98.47	97.29
	LR	98.31	96.13	95.72	99.02	97.99	97.34
FastText (SkipGram)	RF	98.84	97.38	96.79	99.69	98.43	98.22
	DT	98.53	96.69	97.49	98.07	98.76	97.78
	SVM	97.48	94.37	92.41	100	96.36	96.05
	XGBoost	98.46	96.52	96.09	99.24	98.09	97.64
	LR	98.47	96.55	96.54	98.83	98.30	97.67

Table 3.10: Summary of the works executed on all three datasets

Measure (%)	Dataset-1	Dataset-2	Dataset-3
Accuracy	99.50	99.39	99.18
MCC	98.97	98.62	98.15
Precision	99.96	99.19	98.00
TPR	99.20	98.97	99.51
TNR	99.95	99.60	99.02
F-Score	99.58	99.08	98.75
FPR	0.05	0.40	0.98

tested with Word2Vec-SkipGram and RF model. Additionally, From and Message-Id features are able to achieve accuracies of 97.75% and 95.23%, respectively, which are also noteworthy. Thus, it can be inferred that all four selected features are highly effective in classifying emails.

3.7.9 Result Analysis

A series of experiments were conducted using the proposed method to determine the optimal vector sizes. These experiments evaluated the performance of hybrid techniques, which incorporate word embedding and machine learning on individual datasets. The results were organized into tables and analyzed thoroughly to identify the most effective model. According to the results, FastText-CBOW and RF combination achieved the highest accuracy of 99.50% for Dataset-1 with vector size 100. TF-IDF and RF achieved the highest accuracy of 99.39% for Dataset-2 with vector size 200, and the Count Vectorizer and RF achieved the highest accuracy of 99.18% for Dataset-3 with vector size 150. These results show that the Random Forest classifier consistently performs the best with most word embedding algorithms. The achieved results are competitive when we compare with other existing works.

3.7.10 Comparison study

To evaluate the presented approach and determine its effectiveness, a comparison study is conducted using common datasets and methods that have been used in existing research papers. This comparison study is aimed at providing a benchmark against which the performance of the presented approach can be evaluated. The common datasets and methods are chosen to ensure that the comparison is fair and unbiased. By using com-

Table 3.11: Performance of individual features

Feature	Classifier	TF-IDF Accuracy(%)	CV Accuracy(%)	W2V-CBOW Accuracy(%)	W2V-SG Accuracy(%)	FT-CBOW Accuracy(%)	FT-SG Accuracy(%)
From	RF	86.73	86.73	97.72	97.75	94.35	94.11
	DT	86.68	86.60	97.75	97.72	94.35	94.11
	SVM	85.07	83.63	94.30	93.98	74.88	85.50
	XG-Boost	85.74	85.79	96.49	96.60	93.87	93.42
	LR	86.09	86.09	96.01	96.30	87.77	89.94
Message-ID	RF	85.07	85.10	94.29	95.23	90.89	89.92
	DT	85.02	84.99	94.18	95.07	90.91	89.92
	SVM	80.65	79.69	75.54	75.86	53.97	54.37
	XG-Boost	84.91	84.86	93.27	93.83	90.11	89.52
	LR	84.40	84.35	86.33	91.48	51.11	73.02
Return-Path	RF	97.89	97.89	99.33	99.34	98.63	98.61
	DT	97.83	97.86	99.30	99.51	98.63	98.58
	SVM	96.58	96.12	95.94	96.05	86.84	87.42
	XG-Boost	97.60	97.51	98.85	99.19	98.50	98.26
	LR	97.81	97.75	98.40	98.42	93.61	95.11
Subject	RF	96.29	96.20	98.68	98.71	98.16	98.62
	DT	96.26	96.17	97.46	97.46	97.40	97.81
	SVM	94.40	93.67	96.26	96.93	92.68	94.86
	XG-Boost	94.69	94.83	96.58	96.87	96.24	97.34
	LR	95.18	95.12	97.95	98.36	93.58	96.23

Table 3.12: Summary of the works implemented on publicly available datasets.

Author	No. of Features	Type of features	Dataset size	Accuracy (%)
Smadi et al. (2018)	50	Hybrid	Phishing-4559 Ham-4559	98.63
Islam and Abawajy (2013)	21	Hybrid	Unknown	97
Malaysia (2013)	21	Hybrid	Phishing-4300 Ham-6000	99
Khonji et al. (2012)	47	Hybrid	Phishing-4116 Ham-4150	99.37
Gansterer and Pölz (2009)	30	Hybrid	Phishing-5000 Ham-5000	97
Toolan and Carthy (2009)	5	Hybrid	Phishing-4202 Ham-4563	99.31
Hamid and Abawajy (2011)	7	Hybrid	Total-4594	96
Toolan and Carthy (2010)	22	Hybrid	Phishing-4202 Ham-4563	97
Fette et al. (2007)	10	Hybrid	Phishing-860 Ham-6950	96
Proposed work	4	Header	Phishing-9135 Ham-6295	99.50

3. Phishing email detection framework using word embedding and machine learning

mon datasets and methods, it is possible to compare the performance of the presented approach against existing methods and identify its strengths and weaknesses. Additionally, this approach can help in identifying areas where improvements can be made to further enhance the performance of the presented approach. Overall, the comparison study provides valuable insights into the effectiveness of the presented approach and its potential for practical use in real-world applications.

Using common datasets:

This section presents a comparison between the results obtained from our presented work and those obtained from other works that were conducted on the same open-source dataset using common methods. The results from the existing works were extracted directly from their respective papers and are presented alongside our results in Table 3.12. The comparison shows that some of the existing works used multiple hybrid features but achieved lower accuracy compared to our presented work, which only used four header heuristics. It is worth noting that the datasets used by different researchers varied in size, but we used the same publicly available dataset consisting of 9135 phishing emails and 6295 legitimate emails from the open-source corpus. Our presented model outperformed all other existing works by achieving an accuracy of 99.50%.

Using Common methods:

Table 3.13 presents a comparison between the presented work and other word embedding-based techniques. It can be observed that the presented technique outperforms the other techniques with an accuracy of 99.50%, 99.39%, and 99.18% for different datasets. This is a remarkable achievement considering that the presented technique achieved such high accuracy with only four header features of the email. It is worth noting that some of the other techniques used multiple hybrid features and still achieved less accuracy compared to the presented work. The results presented in Table 3.13 further demonstrate the effectiveness and superiority of the presented technique over other existing word embedding-based techniques for email phishing classification.

Table 3.13: Summary of the works that used word embedding techniques

Author	Features	Dataset (s)	Dataset size	Accuracy (%)	Precision (%)	F-score (%)
Nguyen et al. (2018)	Hybrid	IWSPA-AP 2018	Legit:4082 Phish:503	–	99.0	99.1
Bagui et al. (2019)	Hybrid	Private	Legit:14950 Phish:3416	98.89	–	–
Castillo et al. (2020)	Body	Enron, APWG, and Non-public	Legit:84111 Phish:30776	95.68	–	–
Ra et al. (2018)	Body	IWSPA-AP 2018	Legit:5088 Phish:612	99.1	90.59	93.07
Hiransha et al. (2018)	Body	IWSPA-AP 2018	Legit:5088 Phish:612	96.8	–	–
Harikrishnan et al. (2018)	Hybrid	IWSPA-AP 2018	Legit:5088 Phish:612	90.29	92.5	94.6
Verma et al. (2012)	Hybrid	PhishCatch	Legit:1000 Phish:2000	97	–	–
Gutierrez et al. (2018)	Hybrid	Purdue university's Sophos	Legit:158000 Phish:425870	96.5	–	–
Valecha et al. (2021)	Hybrid	Enron Millersmile	Legit:19153 Phish:17902	96.52	98.53	96.31
Presented Model	Header	Dataset - 1	Legit:6295 Phish:9135	99.50	99.96	99.58
		Dataset - 2	Legit:18270 Phish:9135	99.39	99.19	99.08
		Dataset - 3	Legit:18270 Phish:8986	99.18	98.00	98.35

3.8 SUMMARY

This chapter introduces innovative methods for detecting phishing emails by utilizing word embedding and machine learning classifiers. The presented approach uses only four header features of emails for classification. The results of the experiments demonstrate that the FastText-CBOW algorithm combined with RF classification achieves the

3. Phishing email detection framework using word embedding and machine learning

highest accuracy of 99.50% when tested with publicly available datasets. Additionally, the RF classifier performed consistently well with all word embedding algorithms. Therefore, the RF classifier is more appropriate for phishing email classification when used in conjunction with word embedding techniques.

CHAPTER 4

DEEPEPHISHNET: A DEEP LEARNING FRAMEWORK FOR PHISHING EMAIL DETECTION USING WORD EMBEDDING ALGORITHMS

Email phishing refers to a type of social engineering tactic that utilizes fraudulent emails with the aim of deceiving users into revealing their authentic personal or business credentials. Several phishing email detection methods based on machine learning, deep learning, and word embedding have been developed. In this chapter, a deep learning model for detecting email phishing is presented, which employs word embedding algorithms, such as Word2Vec, FastText, and TF-IDF, to represent email messages as vectors. These vectors are then utilized as inputs to a deep neural network, which is trained to classify emails as legitimate or phishing. The effectiveness of the proposed framework is evaluated on a dataset of phishing and legitimate emails, and promising results are obtained. The approach has the potential to enhance the accuracy of email phishing detection systems and reduce the false positive and false negative rates. Notably, the DeepEPhishNet method only employs four header-based features of emails (From, Returnpath, Subject, and Message-ID) as in Chapter 3 for email classification.

4.1 INTRODUCTION

Phishing is a common tactic used by criminals to deceive victims into sharing sensitive information or installing malicious software that can grant access to their networks.

4. DeepEPHishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

What may seem like a harmless email can actually be the beginning of ransomware, cryptojacking, or data theft. This type of attack affects over 100,000 internet users globally every day. Unfortunately, detecting phishing emails has become more challenging over time, as attackers have developed more sophisticated techniques. As humans are often the first line of defense against phishing emails, hackers are using more advanced tactics to trick their targets. With the increase in internet usage, the number of phishing URL and email attacks has also risen, and fraudsters now have access to more resources to carry out complex and risky attacks. Furthermore, the COVID-19 pandemic has provided new opportunities for attackers to disguise their identity and execute phishing attacks. Despite these challenges, it is crucial to stay vigilant and utilize available tools to prevent falling prey to these scams.

The Sophos phishing insights report (Adam 2021) highlights that there has been a significant increase in phishing activities since the start of the pandemic. The report shows that the government sector has experienced a 77% rise in phishing attacks, followed by the professional and business services sector at 76% and the healthcare sector at 73%. Additionally, research by SophosLab has investigated how attackers have profited from the pandemic. They found that attackers have taken advantage of work-from-home arrangements and home package delivery scams. These findings emphasize the need for individuals and organizations to remain vigilant and take proactive measures to protect themselves against phishing attacks.

Cofense's annual state of phishing study, conducted by Higbee (2021), has revealed that email phishing attacks aimed at stealing credentials have had a considerable impact on various industries. The study found that the education sector was targeted the most with a 77% phishing attack rate, followed by retail at 73%, and trade at 71%, among others. The report indicates that criminals have been using various techniques, including mimicking, credential theft, fraudulent calls, and the use of urgency messages, to obtain innocent people's credentials and trap them into financial loss, especially during the peak of the pandemic. These findings emphasize the importance of increased awareness and training to recognize and prevent such attacks. Additionally, implementing strong security measures and maintaining regular updates to software and systems

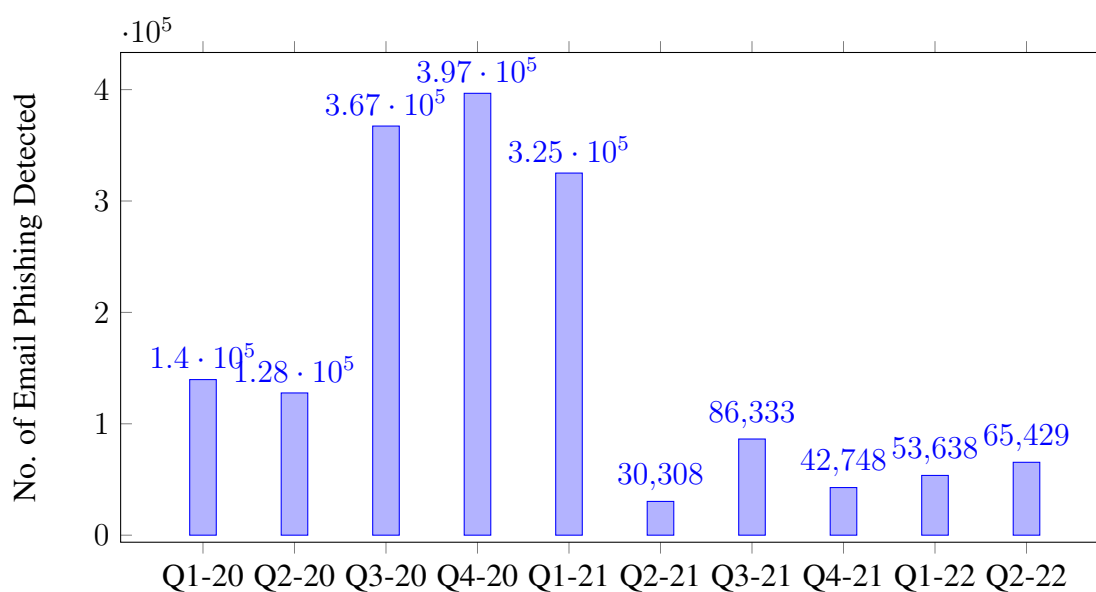


Figure 4.1: APWG 2020-21 Phishing Email Statistics

can help mitigate the risk of credential theft via email phishing.

The APWG’s phishing statistics from Q1-2020 through Q2-2022, as shown in Figure 4.1, reveal that the mid-three quarters of 2020 and 2021 experienced the highest incidences of email fraud. These statistics from the pandemic period prompted researchers to investigate better methods for accurately classifying phishing emails. To address this challenge, the researchers implemented Word Embedding and Deep Learning (WE-DL) techniques. Word embedding creates vector representations of words, which deep learning algorithms can use to classify emails accurately into their respective categories. The WE-DL model produced the most successful results compared to other existing techniques that used a limited number of features. The implementation of these advanced techniques can help improve the accuracy of phishing email detection systems and minimize the risk of individuals falling victim to phishing scams.

This research work has made several significant contributions, including:

- **DeepEPHishNet Framework:** A novel *DeepEPHishNet* framework for email phishing detection using Word Embedding and Deep Learning (WE-DL) techniques has been presented. This framework utilizes Word2Vec, FastText, and TF-IDF to

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

represent email messages as vectors, which are then used as input to a deep neural network trained to classify emails as phishing or legitimate. The WE-DL model successfully identified more phishing emails than other approaches, indicating its potential for improving the accuracy of email phishing detection systems.

- **Improved Detection Accuracy:** DeepEPhishNet framework outperformed existing techniques with a limited number of features and achieved higher accuracy rates in detecting phishing emails.
- **Reduced False Positive and False Negative Rates:** DeepEPhishNet framework also showed a significant reduction in false positive and false negative rates, minimizing the risk of individuals falling victim to phishing scams.
- **Application in Real-World Scenarios:** The DeepEPhishNet framework has the potential for practical application in real-world scenarios, where it can improve the accuracy of email phishing detection systems and help organizations and individuals protect themselves from phishing attacks.

The DeepEPhishNet has the potential to make a valuable contribution to the field of email phishing detection, enhancing the security and protection of individuals and organizations from phishing scams. The presented model for email phishing detection using Word Embedding and Deep Learning (WE-DL) techniques offers several advantages over existing techniques, including:

- **Higher Accuracy:** The model achieved higher accuracy rates in detecting phishing emails, outperforming existing techniques with a limited number of features. This higher accuracy is essential for improving the protection of individuals and organizations against phishing scams.
- **Flexibility:** The WE-DL technique is highly flexible, allowing it to handle a wide range of email data types and formats. This flexibility enables the model to adapt to new types of phishing attacks and improve its accuracy over time.
- **Speed:** The model is fast and efficient, allowing it to process large volumes of

email data quickly. This speed is essential for organizations that need to quickly identify and respond to phishing attacks.

- **Generalizability:** The WE-DL technique is generalizable, meaning that it can be applied to a wide range of email data sets and phishing attack scenarios. This makes the model highly adaptable and applicable to real-world scenarios.

Until this point, there have been no published works that solely rely on email header features for email phishing detection, except for our previous work, which was discussed in Chapter 3. Most phishing detection techniques rely on email content analysis or a combination of content and header information. However, our previous work demonstrated that email header features, such as From, Return-Path, Subject, and Message-ID, can be used effectively to classify emails as either phishing or legitimate. The model in this chapter builds on this previous work by incorporating Word Embedding and Deep Learning techniques to improve the accuracy and efficiency of email phishing detection using only these four header features.

4.2 DEEPEPHISHNET FRAMEWORK

The "*DeepEPhishNet*" model is an advancement of our previous work (Somesha and Pais 2022). The DeepEPhishNet is designed with several stages of email data processing. The first step involves collecting email data to create datasets for evaluating the model. Once the data is collected, the next steps include feature selection, data cleaning, dictionary construction, and vectorization, which were discussed in Chapter 3, section 3.5. However, the classification method used in this chapter is different, as the proposed model employs deep learning classifiers to improve the accuracy and efficiency of email phishing detection. By using word embedding techniques, the model is able to represent email messages as vectors, which are then used as inputs to a deep learning networks for classification. This approach shows promising results and has the potential to improve the accuracy of email phishing detection systems, as well as reduce false positive and false negative rates.

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

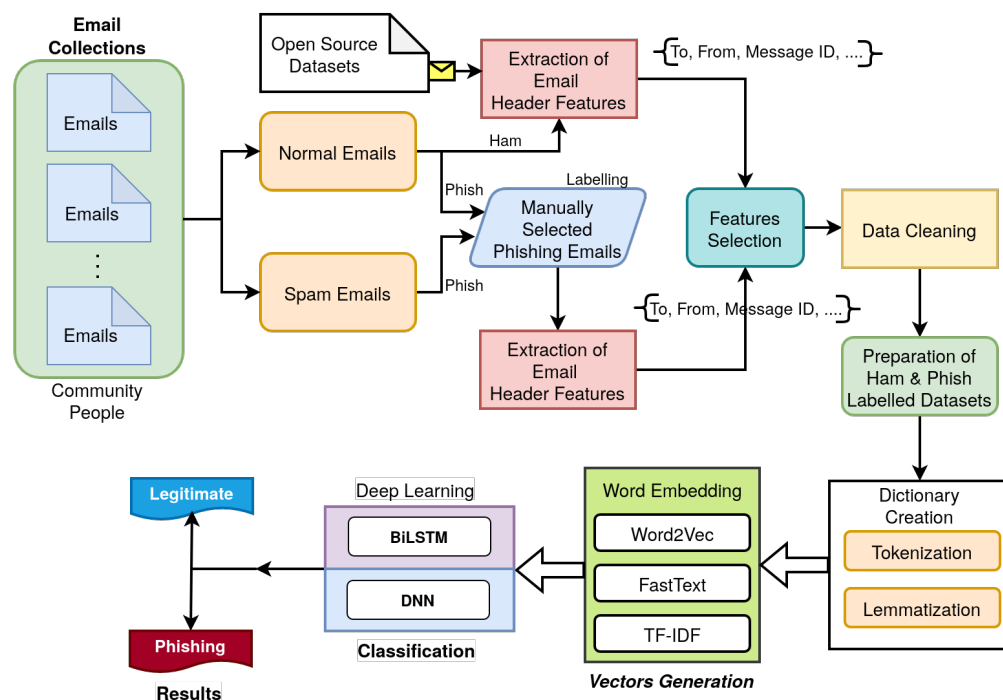


Figure 4.2: Architecture of "DeepEPhishNet" a Phishing Email Classification Framework

4.2.1 Classification

Out of many classification methods used by different researchers discussed in chapter 2, we have selected two of those methods from deep learning models such as Bi-LSTM and DNN. Before using classification, the selected datasets vectors are created by using word embedding techniques such as Word2Vec, FastText, and TF-IDF. These vectors are inputs to the deep learning classifiers to classify given input as phishing or legitimate email by training and testing the model. The used classifiers are discussed below.

Formal Description of LSTM : A recurrent neural network is a specific type of bio-inspired neural network that can model and learn a sequential data pattern. It learns sequential dependencies by learning one sequence at a time and thereby introducing time to neural network modeling.

The recurrent neural network has been good at the sequential and time-series data set and has proved very useful (Bahnsen et al. 2017; Smith and Jin 2014). But the general recurring network suffers from a vanishing gradient problem or an explosive gradient problem, i.e., they can not retain memory across a larger path that causes long-

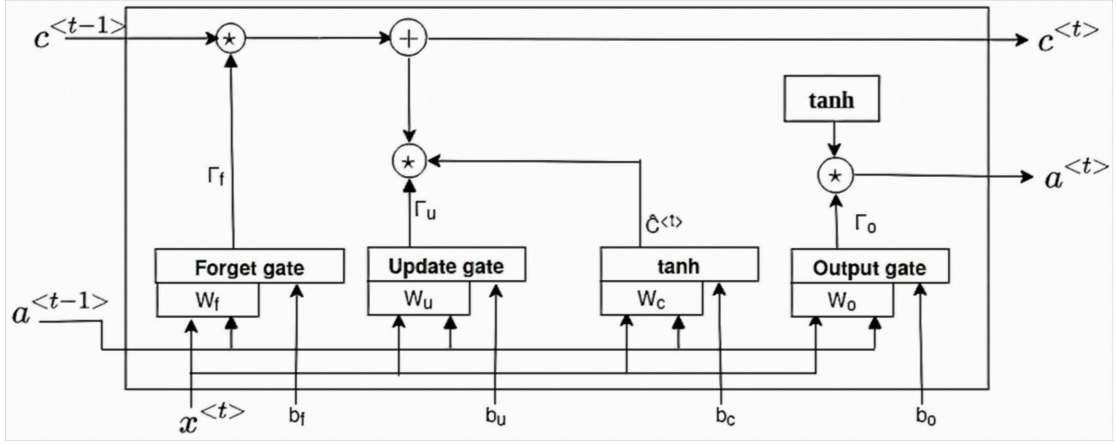


Figure 4.3: Architecture of LSTM

term dependencies (Jozefowicz et al. 2015; Mikolov et al. 2014). And as a result, a long correlation between sequences is not maintained and the network fails in such circumstances. So LSTM takes care of the long correlation between sequences.

LSTM (Figure 4.3) removes the vanishing gradient problem or exploding gradient problem to avoid long term dependencies. In LSTM, a neuron is replaced by cell memory which performs the task using activation function to input by forming a linear combination of the dot product of input and weights with bias. LSTM also, uses update gate, forget gate, and output gate to avoid long term dependencies.

$$\hat{C}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \quad (4.1)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \quad (4.2)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \quad (4.3)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \quad (4.4)$$

$$C^{<t>} = \Gamma_u * \hat{C}^{<t>} + \Gamma_f * C^{<t-1>} \quad (4.5)$$

$$a^{<t>} = \Gamma_o * C^{<t>} \quad (4.6)$$

Weight matrices W_c, W_u, W_f, W_o and bias vector b_c, b_u, b_f, b_o , and temporary cell state ($\hat{C}^{<t>}$), update gate (Γ_u), forget gate (Γ_f), output gate (Γ_o), cell state ($C^{<t>}$) and activation ($a^{<t>}$) respectively (Hochreiter and Schmidhuber 1997) remain same for all time steps in single unit of LSTM network and updated after each epoch during back propagation method. The long memory is usually called cell state (Eq. 4.1). This allows the network to store the information coming from previous cell. It is updated

using the value of both update gate (Eq. 4.2) and forget gate (Eq. 4.3). The forget gates enable the network to forget the information which is not necessary or not relevant by multiplying with 0. It also helps to retain information by multiplying by 1. The update gate determines which information should be entering the cell memory to store. The output gate (Eq. 4.4) decides which result should be moving forward to the next hidden layer. For exploring the feasibility of a recurrent neural network on our data set, we have implemented LSTM.

In this neural network, we have taken four LSTM units performing different mathematical computations as defined earlier. Each unit has 10 timesteps. Each timestep has one output, which is passed as input to the next timestep. The last timestep of the first unit is also passed to the second unit, and so on, and finally, we get output from the last timestep of the fourth unit of LSTM. The obtained output is further densed to a single output and passed it to the sigmoid function. The loss function is calculated and error is optimized using Adam Optimizer. During backpropagation, each parameter of all four units of LSTM is updated. And again, the Loss function is calculated in each epoch; the network learns when variables are updated. We modified the dataset dimensionality to implement LSTM. We have converted 10 features to 10 timesteps, each timestep consists of one feature. Through LSTM, we attempted to find out the possible relationship between different features. Initially, at the first gate, zero vector and first timestep were passed to the first gate. The output from the first LSTM gate passed as input to second, and so on till the tenth gate. The obtained single output from the tenth gate forms a single LSTM unit output, which is again passed as input to the first gate of the second unit. Hence, the output of the previous gate along with the current timestep fed to the next gate and the output of the previous LSTM unit fed to the next unit until the fourth LSTM unit. In the fourth unit, its output was densed to 1, which is passed on to the sigmoid activation function (4.12), and then the loss function (4.13) is calculated and optimized using Adam Optimizer.

Formal Description of Bi-LSTM : Bidirectional LSTMs (Bi-LSTM) are inspired by bidirectional RNNs (Schuster and Paliwal 1997), which use two separate hidden layers to process sequence data in both forward and backward directions. The two

hidden layers are connected to the same output layer by bidirectional LSTMs. It has been demonstrated that bidirectional networks outperform unidirectional networks in many classification applications.

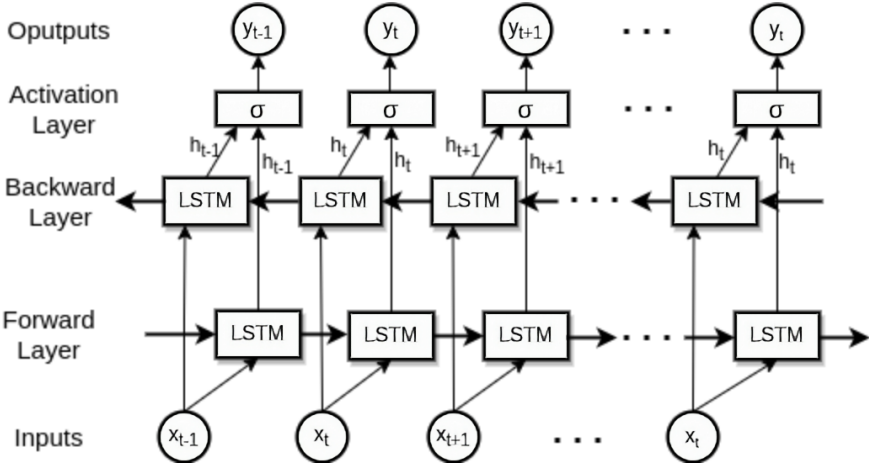


Figure 4.4: Bi-LSTM Architecture

The algorithmic structure of Bi-LSTM has layered structure which contains forward and backward LSTM layers as shown in Figure 4.4. The output of the forward layer h is calculated by using positive sequence inputs iteratively from time $t - n$ to $t - 1$ and backward layer sequence of outputs are calculated using the reverse inputs of time $t - n$ to $t - 1$. The forward layer and backward layer outputs are calculated using the basic LSTM equations Eq. 4.3 to Eq. 4.6. The output of the Bi-LSTM layer is generated as output vector Y_t , and an element of every layer is calculated using the below equation Eq. 4.7:

$$y_t = \sigma(h_t, h_t) \tag{4.7}$$

where the sigmoid function (σ) is used to join the two output sequences together. The output of the Bi-LSTM layer will be represented by a vector $Y_t = (y_{t-n}, \dots, y_t, \dots, y_{t-1})$, where the last element y_{t-1} is predicted speed for next iteration.

The word embedding model takes four selected features from the datasets and functional parameters from Table 4.1 as inputs to generate word vectors. These word vectors are then used as inputs for the Bi-LSTM model. The Bi-LSTM model is designed to classify emails as legitimate or phishing, utilizing specific hyperparameters listed in Table 4.2. The proposed Bi-LSTM model consists of a 300-dimensional output space and

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

includes three LSTM layers (two Bidirectional LSTM layers and one regular LSTM layer), four dropout layers, and four dense layers. The ReLU activation function is applied to the three intermediate layers, while the sigmoid activation function is used at the output layer. The model utilizes the Adam optimizer. The model classification output is '0' for legitimate and '1' for phishing. The network structure of Bi-LSTM model with its layers, shape of the network and selected trainable, non-trainable and total parameters are shown in tabular structure of Figure 4.5.

Table 4.1: Common parameters used in Word2Vec and FastText models

Hyper parameter	Values for Word2Vec	Values for FastText
Test-Train split	30-70	30-70
Vector size	300	300
Window size	10	10
Workers	10	10
Iterations	50	-

Formal Description of DNN : The Deep Neural Network is a type of machine learning technology. It consists of many common neural network layers. It has one input layer, one output layer, and at least one hidden layer as shown in Figure 4.6.

Each layer is composed of the basic computing unit i.e., the neuron. The neuron is inspired by the biological neuron that performs mathematical functions for the storage of information. And this information is transmitted to another neuron and therefore

Table 4.2: Hyper parameters used in Bi-LSTM

Hyper parameter	Values for LSTM
Output dimensionality space	300
Embedding dimension	128
Vocabulary length	100
Output size	256
No. of Epochs	300
No. of LSTM Layers	3 (2-Bi-LSTM, 1-LSTM)
Drop out layers	4
Dense layers	4
Activation functions	ReLU, Sigmoid
Optimizer	Adam

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 1, 512)	4827136
bidirectional_1 (Bidirection)	(None, 1, 256)	656384
lstm_3 (LSTM)	(None, 64)	82176
dropout_1 (Dropout)	(None, 64)	0
dense (Dense)	(None, 32)	2080
activation (Activation)	(None, 32)	0
dropout_2 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dropout_3 (Dropout)	(None, 16)	0
dense_2 (Dense)	(None, 8)	136
dropout_4 (Dropout)	(None, 8)	0
dense_3 (Dense)	(None, 1)	9
Total params: 5,568,449		
Trainable params: 5,568,449		
Non-trainable params: 0		

Figure 4.5: Bi-LSTM Experimental Parameters

information propagates in the neural network. A Neuron's general mathematical representation is:

$$Y^k = \Phi\left(\sum_{i=0}^{i=n} W_i x_i + b\right) \quad (4.8)$$

Where Φ is activation function, $W_i \in R^{L*B}$ is weight of i^{th} neuron and Y^k is the output of i^{th} neuron. The number of neurons in the input layer depends upon the dimension of datasets or equivalent to the number of features of the dataset, i.e., $X \in R^{L*K}$ where L is the total number of the dataset, K is a total number of features in datasets and R represents a real number. The number of neurons in the output layer depends on the number of outputs we want. The number of neurons in the hidden layer is a hyperparameter that needs to be tuned to obtain an optimum result. Since each neuron performs computation, the number of neurons defines the network complexity. Each deep neural network is a complex mathematical function that adapts itself according to the nature of data. So making the network more complex may end up with data

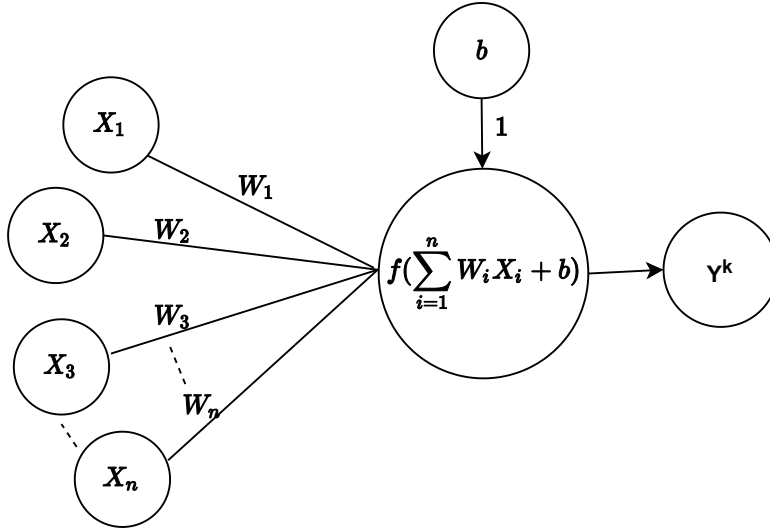


Figure 4.6: Architecture of simple neuron

overfitting, i.e., it performs pretty good with training data but fails to achieve good accuracy with unknown data.

Let $l = \{0, 1, 2, 3, 4, 5\}$ be the layers in my deep learning model. $Y^{(l-1)}$ be the input to layers $\{1, 2, 3, 4, 5\}$, $Y^{(l)}$ be output value of layer, where $W^{(l)}$ be weight of layer i that is used for linear transformation of inputs from n layer to output of m layers. $B^{(l)}$ be bias of layer i , $F^{(l)}$ be the associated activation function to each layers. $Y^{(0)}$ is nothing but input layer and $Y^{(l)}$ is output layer.

$$Z^{(l)} = Y^{(l-1)} * W^{(l)} + B^{(l)} \quad (4.9)$$

$$Y^{(l)} = F(Z^{(l)}) \quad (4.10)$$

where $*$ is for matrix multiplication. W values were initialized with Xavier Initialization (the initializer used to initialize random values) and B initialized with zero. W and B are updated after each iteration in the backpropagation method. Layer 0 is the input layer and output layer 6, layers 1 - 5 are hidden layers activated with the ReLU function provided by:

$$Y_i^l = \begin{cases} 0 & \text{if } Z_i^l \leq 0 \\ Z_i^l & \text{Otherwise} \end{cases} \quad (4.11)$$

where i represent i^{th} iteration and l represent l^{th} layer. The intermediate output of our

model Y^* is obtained from below given sigmoid activation function :

$$Y^* = \frac{1}{1 + \exp -Z^l} \quad (4.12)$$

where $l=6$ in case of output layer. The loss function ($L(Y^*, \hat{Y})$) over entire dataset is defined as sum of cross entropy between model output and actual output, that is shown as below.

$$L(Y^*, \hat{Y}) = \sum_{j=1}^n [\hat{y}_j \log y_j^* + (1 - \hat{y}_j) \log(1 - y_j^*)] \quad (4.13)$$

where Y^* is intermediate output of entire dataset obtained after processing it through deep learning model and $y_j^* \in (0, 1)$ is j^{th} row of Y^* while \hat{Y} is actual label of our dataset and $\hat{y}_j \in \{0, 1\}$ is j^{th} row of \hat{Y} , where 0 represent legitimate site and 1 indicate phishing site. And above given loss function is optimized using Adam Optimizer at every epoch to update parameters and train deep neural model using the backpropagation algorithm. The functional formations represent these features without overfitting, because of DNN has 5 hidden layers along with one input and one output layer.

The proposed DNN model comprises six layers with one input and one output layer as shown in Figure 4.7. In this work, the DNN model takes an output vectors from the word embedding model as input. Along with the vectors, the model requires various hyperparameters, including the number of nodes, batch size, epochs, number of hidden layers, dropout layer, output layer, activation functions, and optimizer. The model utilizes 264 nodes, a batch size of 64, ReLU activation for all layers except the output layer, and sigmoid activation for the output layer. The ReLU and sigmoid activation functions are used to standardize hidden and output layers. The output of the model Y^k is '0' for legitimate and '1' for phishing. The rational for dropout is to speed up the training by decreasing the internal covariate shift and overfitting. The specific hyperparameters used are presented in Table 4.3, and the layered structure, output shape and parameters used in each layer of DNN network is illustrated in tabular format Figure 4.8.

4.3 EXPERIMENTAL EVALUATION

This section discusses the basic experimental setup, resources used, and evaluation metrics used to evaluate the DeepEPhishNet model. In this work, we have used email as

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

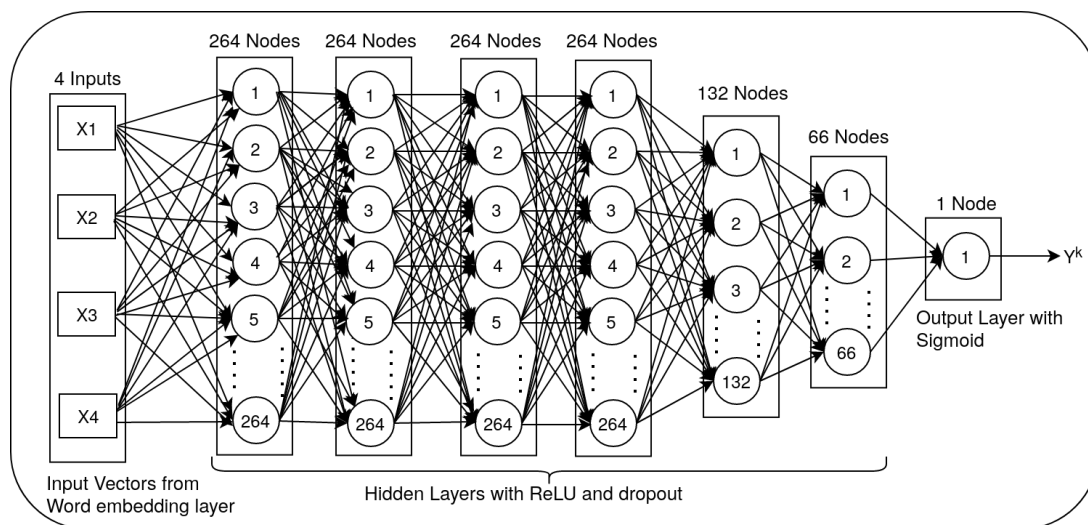


Figure 4.7: DNN Architecture

Table 4.3: Hyper parameters used in DNN

Hyper parameters	Values for DNN
Layer size	264
Batch size	64
No. of Epochs	300
Dense layers	6
Drop out layers	6
Output layer	1
Activation layers	6-ReLU, 1-Sigmoid
Optimizer	Adam(lr=3e-4)

the basic resource. The required emails are gathered from two different resources, one from a predefined open source corpus and another from an in-house generated corpus as discussed in Chapter 2. The dataset preparation procedure is also discussed in Chapter 2, section 2.3.1, and the prepared datasets with corpus size are tabulated in Table 2.8. The evaluation metrics used are also discussed in chapter 2 section 2.2.

4.4 RESULTS AND DISCUSSION

Various experiments were conducted to assess the effectiveness of the proposed model on three datasets, namely Dataset-1, Dataset-2, and Dataset-3. The experiments were carried out in five stages.

In the first experiment, we evaluated the performance of the Word2Vec-SkipGram

Layer (type)	Output Shape	Param #
first_layer (Dense)	(None, 264)	1109064
dropout_1 (Dropout)	(None, 264)	0
second_layer (Dense)	(None, 264)	69960
dropout_2 (Dropout)	(None, 264)	0
third_layer (Dense)	(None, 264)	69960
dropout_3 (Dropout)	(None, 264)	0
fourth_layer (Dense)	(None, 264)	69960
dropout_4 (Dropout)	(None, 264)	0
fifth_layer (Dense)	(None, 132)	34980
dropout_5 (Dropout)	(None, 132)	0
sixth_layer (Dense)	(None, 66)	8778
dropout_6 (Dropout)	(None, 66)	0
class (Dense)	(None, 1)	67
Total params: 1,362,769		
Trainable params: 1,362,769		
Non-trainable params: 0		

Figure 4.8: DNN Experimental Parameters

model using Bi-LSTM and DNN over the three datasets. The second experiment focused on evaluating the performance of the Word2Vec-CBOW model using Bi-LSTM and DNN over the same datasets. For the third and fourth experiments, we evaluated the performance of the FastText-Skipgram and FastText-CBOW models, respectively. Finally, in the fifth experiment, we evaluated the performance of the TF-IDF model using Bi-LSTM and DNN. The basic experimental setup is provided below.

4.4.1 Basic experimental setup

Python was the programming language used for the proposed work, along with common libraries such as *gensim*, *pandas*, *nlTK*, *sklearn*, *numpy*, *TensorFlow*, and *Keras*. The neural network framework was developed using Google Colaboratory¹.

¹<https://colab.research.google.com/>

The work employed three word embedding algorithms, namely Word2Vec, FastText, and TF-IDF. Table 4.1 lists the common parameters used for experiments 1 to 4 for all three word embeddings. The deep learning classifiers utilized in the proposed work were Bi-LSTM and DNN. Tables 4.2 and 4.3 present the basic functional parameters and their corresponding values for these deep learning classifiers.

4.4.2 Experiment-1: Evaluation of Word2Vec-SkipGram model

For this experiment, we have tabulated the functional parameters used to obtain the vectors in Table 4.1. We have used the gensim library, applied cosine similarity, and set the parameter *word2vec – skipgram* to 1. These parameters have been selected after conducting several experiments by varying the values of all parameters. This model has generated a comprehensive set of vectors for the selected parameters, which are then used as inputs to the deep learning models such as Bi-LSTM and DNN to check the legitimacy of the emails.

To accomplish this, the Bi-LSTM model has been initialized with its functional parameters, as mentioned in Table 4.2, and the neural network layers illustrated in Figure 4.5. In the current architecture, we have used three LSTM layers, out of which two are Bi-LSTM layers and one is an LSTM layer. Additionally, we have used four dropout, four dense, and one activation layer. The activation functions used are ReLU at three intermediate layers and *sigmoid*(σ) at the output layer. Finally, we have used the Adam optimizer.

The parameters utilized for constructing the DNN ensemble network with the embedding model are presented in Table 4.3 and Figure 4.8. The ReLU activation function is used with all dense layers, except for the output layer where the *Sigmoid* (σ) activation function is used. Each dense layer has a size of 264, except for the last dense layer which only has one layer. Using these network setup properties, the model's performance is evaluated and presented in Table 4.4 for all three datasets. Based on the data presented in Table 4.4, it can be seen that the model ensemble including a DNN achieves the highest accuracy levels. Specifically, for Dataset-1, the accuracy rate is 99.14%, while for Dataset-2, the accuracy rate using the Bi-LSTM model is 98.74%, and for Dataset-3, the DNN model achieves an accuracy rate of 99.43%. Notably, the

Table 4.4: Performance of Word2Vec-SkipGram model

Dataset	Classifiers	TPR	TNR	F-score	Precision	MCC	Validation Loss	Validation Accuracy
Dataset-1	Bi-LSTM	99.37	98.93	98.93	98.49	98.18	0.0088	99.11
	DNN	99.45	98.93	98.91	98.37	98.28	0.0083	99.14
Dataset-2	Bi-LSTM	99.59	97.10	99.04	98.50	97.20	0.0125	98.74
	DNN	99.32	97.39	99.00	98.37	98.69	0.0123	98.67
Dataset-3	Bi-LSTM	99.16	99.77	99.54	99.92	98.17	0.0068	99.31
	DNN	99.29	99.89	99.62	99.96	98.49	0.0057	99.43

DNN model is particularly effective for Dataset-3, which is an in-house dataset. Additionally, it is worth mentioning that DNN models tend to perform better than Bi-LSTM models when using the Word2Vec-SkipGram model. Table 4.4 also includes other evaluation parameters such as TPR, TNR, F-score, Precision, MCC, and Validation loss.

4.4.3 Experiment-2: Evaluation of Word2Vec-CBOW model

In this experiment, we utilized the Word2Vec-CBOW model for vectorization after selecting functional parameters, similar to experiment 1, with the exception that *word2vec-skipgram* was set to 0 (while *word2vec-cbow* was set to 1). The output of this model was then used as input to both the DNN and Bi-LSTM models, using the same models and functional parameters outlined in experiment 1. The results of this experiment are shown in Table 4.5, where the DNN model achieved the highest accuracy rates of 99.18% for Dataset-1, 98.74% for Dataset-2, and 99.31% for Dataset-3. It is worth noting that the DNN model was particularly well-suited for classification using the Word2Vec-CBOW model, and achieved the highest accuracy rate of 99.31% for the in-house dataset (Dataset-3). Additional evaluation parameters are also presented in Table 4.5.

Table 4.5: Performance of Word2Vec-CBOW model

Dataset	Classifiers	TPR	TNR	F-score	Precision	MCC	Validation Loss	Validation Accuracy
Dataset-1	Bi-LSTM	99.94	98.49	98.78	97.65	98.02	0.0095	99.05
	DNN	99.67	98.86	98.96	98.26	98.29	0.0082	99.18
Dataset-2	Bi-LSTM	99.55	96.18	98.77	98.00	96.43	0.0153	98.38
	DNN	99.38	97.47	99.05	98.73	97.16	0.0112	98.74
Dataset-3	Bi-LSTM	99.10	99.77	99.51	99.92	98.04	0.0073	99.27
	DNN	99.28	99.40	99.54	99.80	98.16	0.0068	99.31

4.4.4 Experiment-3: Evaluation of FastText-SkipGram model

The functional parameters used to obtain the vectors in this experiment are presented in Table 4.1, and an additional parameter, *fasttext – skipgram*, was set to 1, following the same approach as experiment 1. These parameters and their associated values were selected after conducting several experiments, and the FastText-SkipGram embedding model was utilized to generate efficient vectors. The output vector of the model is the input to deep learning classifiers as described in experiment 1. The results of the ensemble model are presented in Table 4.6, where the DNN model achieved the highest accuracy rates of 98.90% for Dataset-1, 98.74% for Dataset-2 with the Bi-LSTM model, and 99.52% for Dataset-3 with the DNN model. It is worth noting that the DNN model was particularly effective for the in-house dataset (Dataset-3), achieving the highest accuracy rate of 99.52%. Additionally, the DNN model outperformed the Bi-LSTM models when utilizing the FastText-SkipGram model. Table 4.6 also includes the results of other evaluation parameters.

Table 4.6: Performance of FastText-SkipGram model

Dataset	Classifiers	TPR	TNR	F-score	Precision	MCC	Validation Loss	Validation Accuracy
Dataset-1	Bi-LSTM	99.19	96.87	98.78	98.39	96.43	0.0147	98.39
	DNN	99.50	98.52	98.59	97.70	97.70	0.0110	98.90
Dataset-2	Bi-LSTM	99.19	96.87	98.78	98.39	96.43	0.0147	98.39
	DNN	99.20	96.15	98.60	98.00	95.90	0.0163	98.15
Dataset-3	Bi-LSTM	99.52	99.14	99.62	99.71	98.46	0.0057	99.42
	DNN	99.38	99.92	99.68	99.97	98.71	0.0046	99.52

4.4.5 Experiment-4: Evaluation of FastText-CBOW model

In this particular experiment, we employed the FastText-CBOW model for vectorization, utilizing the same functional parameters as in experiment 3, with the only difference being that *fasttext – skipgram* was set to 0, and *fasttext – cbow* was set to 1. The resulting vectors generated from this model were fed as input to Bi-LSTM and DNN models, using the same LSTM and DNN models utilized in previous experiments. The outcomes obtained from the deep learning classifiers were collected and tabulated in Table 4.7, where the DNN model consistently achieved the highest accuracy rates for all three datasets. From the results, it was evident that the DNN model attained the high-

est accuracy of 98.81% for Dataset-1, 98.26% for Dataset-2, and 99.48% for Dataset-3. The DNN model is determined to be the optimal choice for classifying phishing emails based on the FastText-CBOW model. Furthermore, it is essential to mention that the DNN model achieved the highest accuracy rate for the in-house dataset (Dataset-3), which was 99.48%. The other evaluation parameters have also been included in Table 4.7.

Table 4.7: Performance of FastText-CBOW model

Dataset	Classifiers	TPR	TNR	F-score	Precision	MCC	Validation Loss	Validation Accuracy
Dataset-1	Bi-LSTM	97.26	99.11	98.24	97.26	97.02	0.0124	98.57
	DNN	99.50	98.38	98.48	97.48	97.52	0.0113	98.81
Dataset-2	Bi-LSTM	99.14	95.85	98.49	97.84	95.58	0.0179	98.01
	DNN	99.37	96.17	98.68	98.00	96.15	0.0155	98.26
Dataset-3	Bi-LSTM	97.71	100	98.84	100	95.32	0.0176	98.24
	DNN	99.42	99.66	99.65	99.89	98.61	0.0050	99.48

4.4.6 Experiment-5: Evaluation of TF-IDF model

The TF-IDF model is utilized in this study to create word vectors for the email header data inputs. The functional parameters for generating these vectors include a random test-train split of 30-70% of the dataset size, a selected random state of 2, and a maximum feature count of 100. The vectors produced are then fed into deep learning classifiers, including Bi-LSTM and DNN, as discussed in previous experiments. Table 4.8 presents the results, which show that the DNN model outperformed the Bi-LSTM model with an accuracy of 98.81% for Dataset-1, 98.65% for Dataset-2 (with the Bi-LSTM model), and 99.04% for Dataset-3 (with the DNN model). Notably, the DNN model attained the highest accuracy of all models for the in-house dataset (Dataset-3). Additionally, the DNN model achieved better accuracy than the Bi-LSTM models for the TF-IDF model. Table 4.8 also provides details on other evaluation parameters.

4.4.7 Models performance analysis with individual datasets

The present work employed word embedding techniques and deep learning models on three datasets, and their performance was evaluated using bar charts depicted in Figures 4.9, 4.10, and 4.11. The analysis indicated that the combination of Word2Vec-CBOW with DNN yielded the best performance on Dataset-1, achieving an impressive accuracy

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

Table 4.8: Performance of TF-IDF model

Dataset	Classifiers	TPR	TNR	F-score	Precision	MCC	Validation Loss	Validation Accuracy
Dataset-1	Bi-LSTM	98.94	98.27	98.12	97.32	96.93	0.0125	98.53
	DNN	99.22	98.55	98.48	97.76	97.51	0.0104	98.81
Dataset-2	Bi-LSTM	98.31	99.76	99.11	99.92	96.39	0.0134	98.65
	DNN	99.11	96.77	98.70	98.30	96.23	0.0132	98.30
Dataset-3	Bi-LSTM	98.30	99.76	99.11	99.92	96.41	0.0135	98.65
	DNN	98.93	99.39	99.36	99.80	97.45	0.0092	99.04

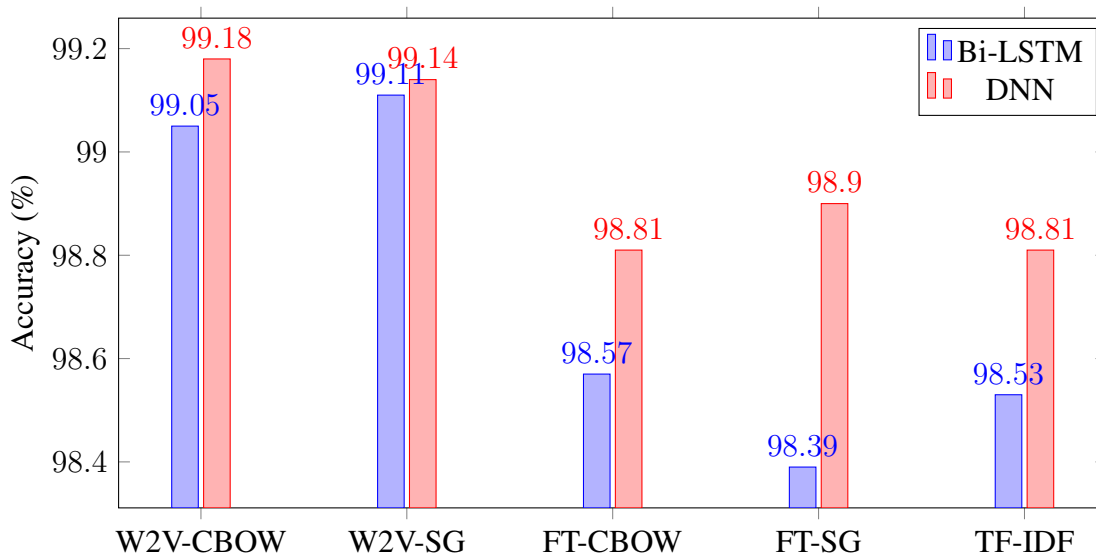


Figure 4.9: Performance of the Model with Dataset-1

of 99.18%. On Dataset-2, both Word2Vec-CBOW with DNN and Word2Vec-SkipGram with Bi-LSTM models exhibited similar and commendable accuracy levels of 98.74%. Likewise, for Dataset-3, all models utilizing DNN showed strong performance, with FastText-SkipGram with DNN achieving the highest accuracy of 99.52%. The findings of the analysis suggested that most word embedding techniques demonstrated enhanced performance when used in conjunction with the DNN algorithm. The validation accuracy and validation loss charts for the best performing model with individual datasets were shown in Figure 4.12, 4.13, & 4.14. Furthermore, the confusion matrix for the best overall results from individual datasets was tabulated in Table 4.9.

4.4.8 Discussion of Bi-LSTM and DNN results

Table 4.10 provides a summary of the results obtained from using different word embedding techniques to detect phishing emails with Bi-LSTM and DNN. As shown in

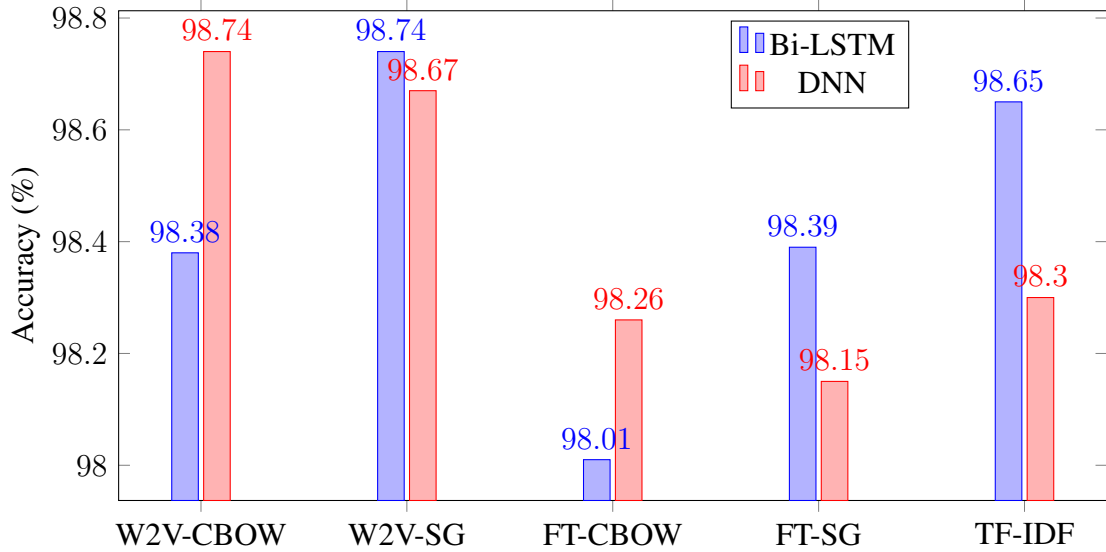


Figure 4.10: Performance of the Model with Dataset-2

Table 4.9: Confusion Matrix (a). Dataset-1, (b). Dataset-2, (c). Dataset-3

	True Positive	True Negative		TP	TN		TP	TN
False Positive	2741	1	FP	2701	22	FP	2652	54
False Negative	22	1865	FN	28	5471	FN	13	5458
	(a)			(b)			(c)	

the table, DNN demonstrates better performance than Bi-LSTM in the classification of phishing emails. Specifically, using FastTextSkipGram on Dataset-3, the DNN model achieves the highest accuracy of 99.52%, compared to 99.42% for the Bi-LSTM model. The lowest accuracy achieved by the DNN model is 98.15% with FastText-SkipGram on Dataset-2, whereas the Bi-LSTM model's lowest accuracy is 98.01% with FastText-CBOW on Dataset-2. Therefore, it can be concluded that DNN models with word embedding techniques are more suitable for phishing email classification.

4.4.9 Result Analysis

The analysis of the performance of various word embedding and deep learning models reveals that the DNN classifier consistently outperforms the Bi-LSTM classifier across all three datasets. It is observed from the achieved results tabulated in Tables 4.4, 4.5, 4.6, 4.7, & 4.8, and shown in Figures 4.9, 4.10, 4.11, 4.12, 4.13, and 4.14 that DNN

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

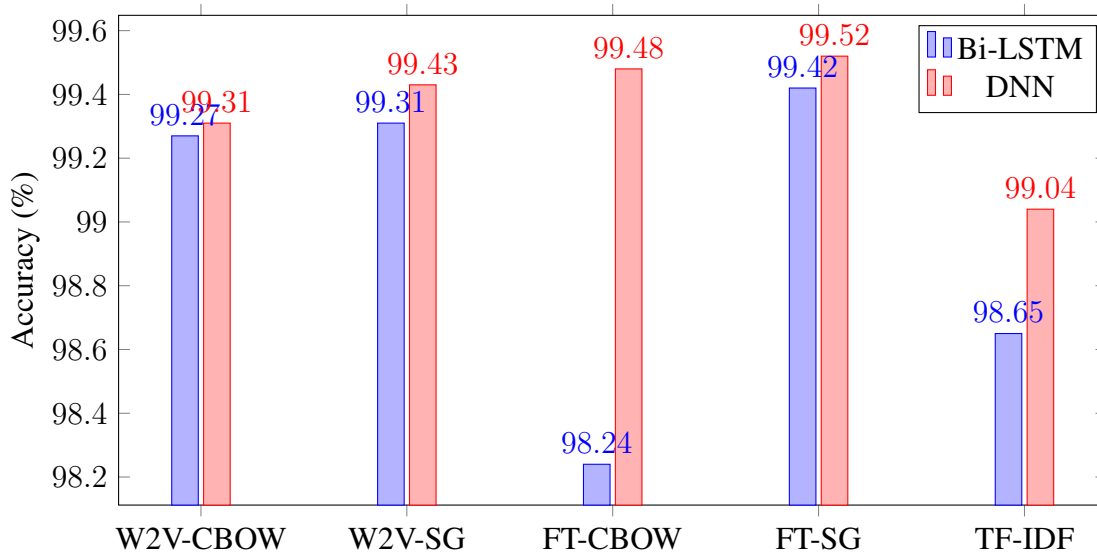
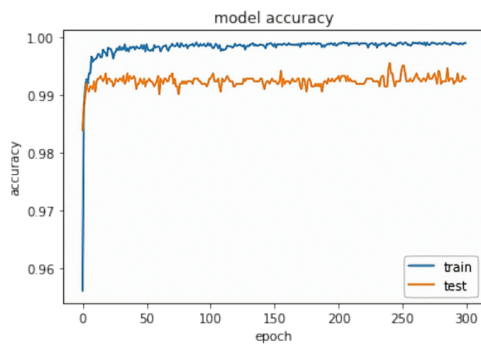
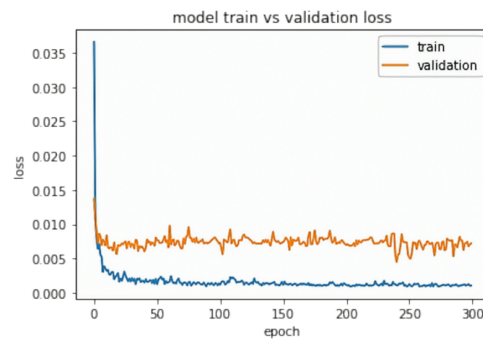


Figure 4.11: Performance of the Model with Dataset-3

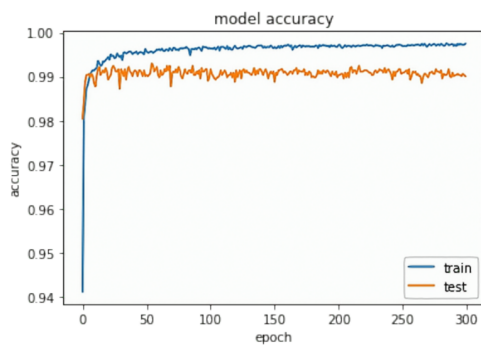


(a) Model accuracy

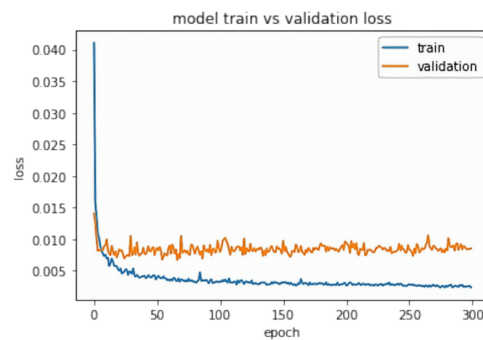


(b) Model loss

Figure 4.12: Validation Accuracy & Loss graphs - Word2Vec-CBOW cum DNN model with Dataset-1



(a) Model accuracy



(b) Model loss

Figure 4.13: Validation Accuracy & Loss graphs - Word2Vec-CBOW cum DNN model With Dataset-2

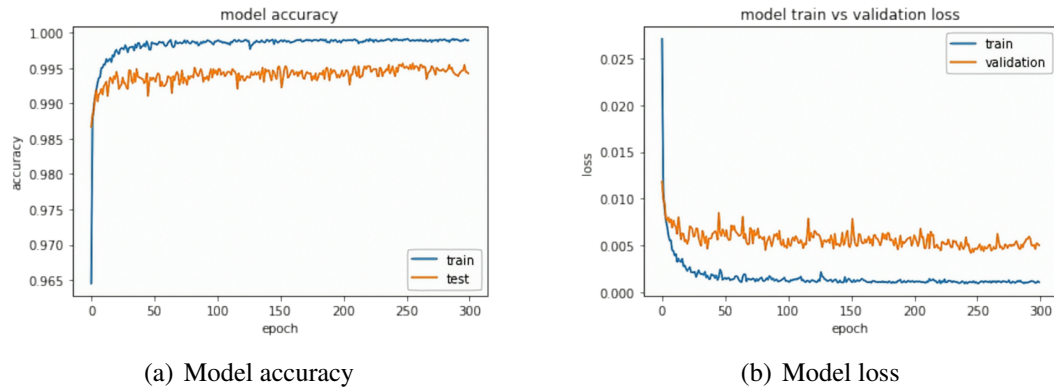


Figure 4.14: Validation Accuracy & Loss graphs - FastText-SkipGram cum DNN model With Dataset-3

consistently exhibiting good results than Bi-LSTM.

Table 4.10: Summary of Bi-LSTM and DNN results with word embeddings

Dataset	Word Embedding	Bi-LSTM	DNN
		Accuracy	Accuracy
1	Word2Vec - CBOW	99.05	99.18
	Word2Vec-Skipgram	99.11	99.14
	FastText - CBOW	98.57	98.81
	FastText-Skipgram	98.39	98.90
	TF-IDF	98.53	98.81
2	Word2Vec - CBOW	98.38	98.74
	Word2Vec-Skipgram	98.74	98.67
	FastText - CBOW	98.01	98.26
	FastText-Skipgram	98.39	98.15
	TF-IDF	98.65	98.30
3	Word2Vec - CBOW	99.27	99.31
	Word2Vec-Skipgram	99.31	99.43
	FastText - CBOW	98.24	99.48
	FastText-Skipgram	99.42	99.52
	TF-IDF	98.65	99.04

4.4.10 Comparison with existing works

Table 4.11 presents a summary of prior studies that have utilized word embedding in conjunction with deep learning techniques for phishing email classification. None of the studies considered using header features alone for classification; rather, they relied on body features or a combination of header and body features. The best accuracy achieved among these studies was 99.1%, which was obtained using email body features

4. DeepEPhishNet: A Deep learning framework for phishing email detection using Word embedding algorithms

as reported by Ra et al. (2018). In contrast, our model achieved a higher accuracy of 99.52% for our in-house dataset (Dataset-3) using only four email header features. This surpasses the performance of previous models, demonstrating the effectiveness of our approach.

Table 4.11: Existing works comparison

Author	Features	Dataset (s)	Dataset size	Accuracy (%)	Precision (%)	F-score (%)
Li et al. (2020)	Hybrid	Private	Large Dataset	95	~94	~94
Nguyen et al. (2018)	Hybrid	IWSPA-AP 2018	Legit:4082 Phish:503	–	99.0	99.1
Bagui et al. (2019)	Hybrid	Private	Legit:14950 Phish:3416	98.89	–	–
Castillo et al. (2020)	Body	Enron, APWG, and Non-public	Legit:84111 Phish:30776	95.68	–	–
Ra et al. (2018)	Body	IWSPA-AP 2018	Legit:5088 Phish:612	99.1	90.59	93.07
Hiransha et al. (2018)	Body	IWSPA-AP 2018	Legit:5088 Phish:612	96.8	–	–
Proposed Model	Header	Dataset - 1	Legit:6295 Phish:9135	99.18	98.26	98.96
		Dataset - 2	Legit:18270 Phish:9135	98.74	99.05	98.73
		Dataset - 3	Legit:18270 Phish:8986	99.52	99.68	99.97

4.5 SUMMARY

This study introduces a new model named DeepEPhishNet, which combines deep learning and word embedding techniques to classify phishing emails. To evaluate the effectiveness of the proposed model, an in-house email dataset was created. The results demonstrated that word embedding, when used with DNN models, achieved the highest accuracy of 99.52% for the in-house dataset, indicating the robustness of our approach. Additionally, the study found that DNN models outperformed Bi-LSTM models for the classification of phishing emails. The ability of our models to achieve high accuracy using only four email header features for classification underscores their simplicity and effectiveness in identifying phishing emails. Overall, the proposed DeepEPhishNet

model provides a promising solution for accurately detecting phishing emails, which can help protect users from cyber attacks.

CHAPTER 5

PHISHING CLASSIFICATION BASED ON TEXT CONTENT OF AN EMAIL BODY USING TRANSFORMERS

Phishing email classification using email header features with word embedding and machine learning is discussed in chapter 3, and word embedding with deep learning is discussed in chapter 4. The results obtained in these two chapters are comparative. Further to re investigate, we used email body text as an input to the deep learning model. The deep learning model used in the current work is BERT transformer to analyze the text content of the email body. BERT, or Bidirectional Encoder Representations from Transformers, is a state-of-the-art natural language processing pre-training technique that uses deep learning to understand the context of a given sentence. Using BERT, the text content of the email body can be analyzed to determine whether it is legitimate or phishing. The BERT transformer will look at the words used in the email body, as well as the grammar and syntax, and will then classify the email as either legitimate or phishing. The advantage of using BERT for email classification is that it enables the system to understand the context of the email body and accurately detect phishing emails. It is also more accurate and efficient than traditional methods, such as using keyword searches, which can often result in false positives.

To summarize, using BERT transformers for phishing classification is an effective and efficient way to detect and classify emails as phishing or legitimate.

5.1 INTRODUCTION

Phishing attacks pose a significant threat to individuals and organizations, targeting sensitive information through deceptive emails. Detecting and preventing such attacks is of paramount importance in ensuring cybersecurity. In recent years, deep learning models, particularly Transformers, have emerged as powerful tools in natural language processing tasks. This chapter focuses on exploring the application of Transformers in classifying phishing emails based on the textual content of the email body. The text content of an email body contains valuable information that can help identify phishing attempts. By leveraging the capabilities of Transformers, which excel in capturing contextual information and long-range dependencies, we aim to build a robust phishing classification system.

In this chapter, we delve into the theoretical foundations of Transformers and their effectiveness in understanding and processing textual data. We discuss the specific challenges posed by phishing emails and how Transformers can address these challenges by modeling the intricate relationships between words and phrases. The primary objective of this study is to design and implement a phishing classification framework using Transformers. We investigate BERT Transformer architectures and explore the performance in detecting and differentiating between legitimate emails and phishing emails. To evaluate the effectiveness of our proposed framework, we utilize a comprehensive dataset consisting of labeled email bodies, encompassing both legitimate and phishing instances. We conduct rigorous experiments and assess the performance of the BERT Transformer model in terms of accuracy, precision, recall, and F1 score. Furthermore, we discuss the interpretability of the Transformer-based phishing classification system, providing insights into how the model makes its predictions and identifying important features for distinguishing between legitimate and phishing emails.

The researchers have proposed email phishing detection algorithms based on supervised learning and unsupervised learning. These algorithms use different machine learning and deep learning algorithms for the classification. The existing methods make use of hybrid features (Body and Header) for the classification (A Hamid and Abawajy 2011; Abu-Nimeh et al. 2009; Bagui et al. 2019; Gansterer and Pölz 2009; Harikr-

ishnan et al. 2018; Islam and Abawajy 2013; Khonji et al. 2012; Ma et al. 2009; Nguyen et al. 2018; Ra et al. 2018; Smadi et al. 2018; Toolan and Carthy 2009; Valecha et al. 2021). Some works have used email header features only for the detection of phishing email (Somesha and Pais 2022). Some researchers (101; Bountakas et al. 2021; Castillo et al. 2020; Hiransha et al. 2018; Ramanathan and Wechsler 2012) have used only the body part of the email for the phishing email detection. In the current work, we are presenting a novel technique based on transformers that uses only the email body text content.

Overall, this chapter aims to contribute to the field of email security by showcasing the potential of Transformers in phishing classification based on the text content of an email body. By harnessing the power of deep learning and Transformer models, we strive to enhance the detection and mitigation of phishing attacks, ultimately bolstering cybersecurity.

The following are the contributions of this work:

- The presented work describes a novel deep learning technique that uses transformers to identify phishing emails using email body text as input.
- The model performance is evaluated on open source and in-house generated datasets.
- Finally, a comparison study is performed on our proposed model and other existing works.

5.2 BERT BASED PHISHING DETECTION

The architecture and its functional parameters of the work is discussed in this section. The architecture is designed to satisfy the objective of the work as shown in Figure 5.1. The objective of the work is to classify the email as phishing or ham using only the text part of an email body. The proposed architecture uses the following steps such as email collection, in-house dataset preparation, data pre-processing and training and classification using transformers.

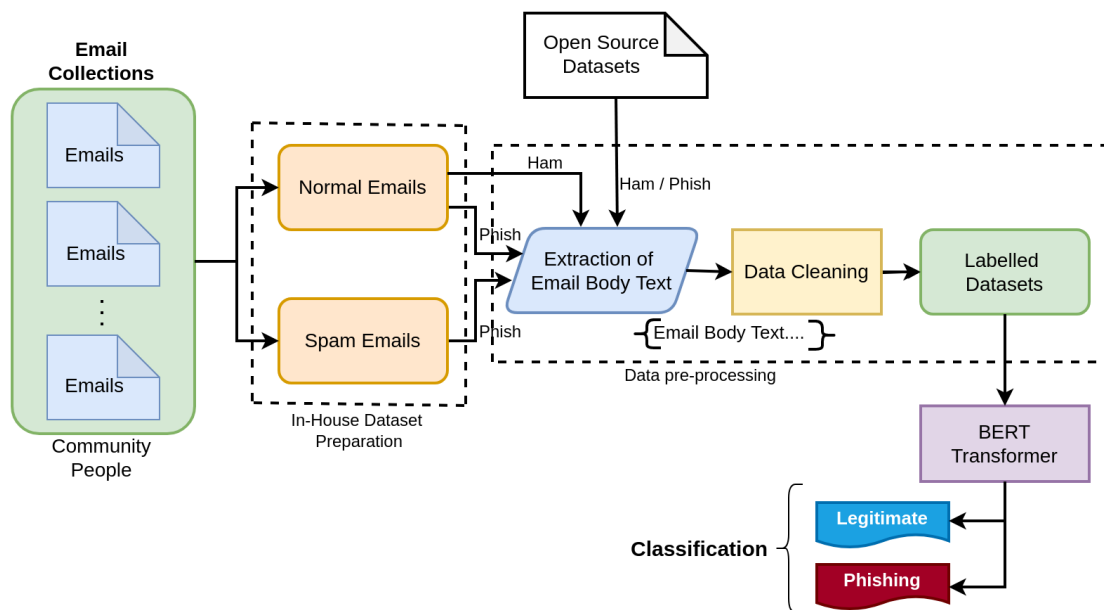


Figure 5.1: Architecture of the model

5.2.1 Emails collection

Email datasets serve as essential inputs for classifying emails as either legitimate or phishing. Several open-source datasets, such as Nazario, SpamAssassin, Enron, IWSPA, and CLAIR-ACL, are widely available, and researchers often rely on them to demonstrate the efficacy of their methodologies. However, some of these open-source datasets have become outdated and no longer reflect the current landscape of phishing techniques. Intruders have become knowledgeable about these datasets and have devised new tricks and techniques to evade detection. Phishers constantly develop fresh approaches to deceive innocent users and gain financial benefits through the acquisition of account credentials. To combat these fraudulent phishing activities, anti-phishing techniques have been introduced to address these evolving tactics. Real-time phishing and legitimate emails are fundamental resources necessary for constructing tools aimed at effectively combating these new tricks and techniques. In our research, detailed in Chapter 3, we have created proprietary datasets sourced from personal emails of institution students, family members, and friends. These emails have been meticulously collected, analyzed, and curated to establish both phishing and legitimate datasets, ensuring their relevance and alignment with current phishing practices.

5.2.2 In-house dataset preparation

In-house datasets are prepared by analyzing the behavior of the recently collected emails, source code of the original emails, Google warning indicators, and using MxToolbox¹ online tool discussed in chapter 2. The prepared in-house dataset-III and its size is tabulated in Table 5.1.

5.2.3 Open source dataset collection

The phishing data for the open-source dataset was collected from the Nazario² repository. This dataset encompasses emails spanning the years 2004 to 2017. The legitimate email data, on the other hand, was sourced from the SpamAssassin³ repositories. These repositories provide ham data collected during the period from 2002 to 2004. It's worth noting that the open-source datasets may not align with the timeframe in which they were collected, potentially rendering them susceptible to being learned by phishers. To address the issue of period mismatch, we have curated Dataset-II, which combines legitimate emails from our in-house corpus captured between 2004 and 2017, along with phishing emails sourced from the Nazario phishing corpus. The creation of Dataset-II aims to mitigate the problem of mismatched timeframes and enhance the relevance and effectiveness of the dataset for our research purposes.

5.2.4 Data pre-processing

Initially, the input data should be preprocessed using Python scripts. Python scripts are written to process emails collected from open source and in-house repositories as input. The developed Python scripts extract the body text of an email from MBOX files and remove unwanted tags, junk, and special characters. Cleaning the processed data involves removing inflectional endings and special characters from the email body text.

5.2.5 Training and classification using transformers

A transformer is a deep learning model used majorly in the fields of NLP and computer vision. The transformer is introduced by Vaswani et al. (2017) and designed to process

¹<https://mxtoolbox.com/Public/Tools/>

²<https://monkey.org/~jose/phishing/>

³<https://spamassassin.apache.org/old/publiccorpus/>

5. Phishing Classification based on Text Content of an Email Body using Transformers

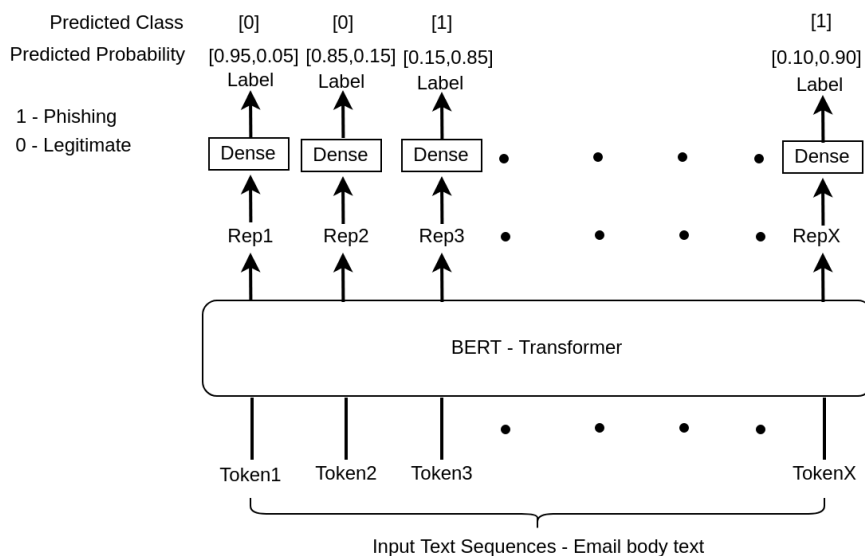


Figure 5.2: BERT - Transformer architecture

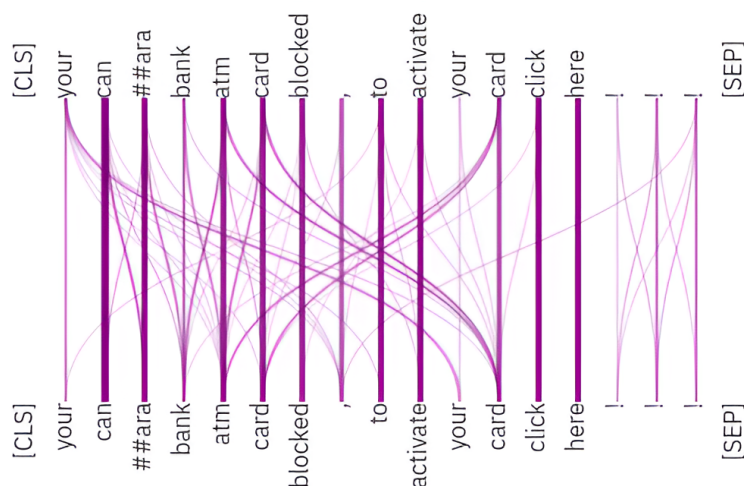


Figure 5.3: BERT base uncased - Example

sequential data for translation and text summarization by using an attention mechanism. Among many transformer models, Bidirectional Encoder Representations from Transformers (BERT) is one of the popular and efficient language transformation models proposed by Google and published by Devlin et al. (2018) and his colleagues⁴ as a new language representation model.

In this work, we used BERT model, a popular NLP model to classify emails by training only body text which is in the form of natural language. BERT uses bi-directional

⁴<https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>

context word learning in left to right and right to left contexts of email content. To do language processing and classification, we use a bert-base-uncased pre-trained model is of Huggingface’s library called transformers to train and classify the given email body text as shown in Figure 5.2. The model includes 12 transformer layers, 768 hidden sizes, 12 self-attention heads, and 110 million parameters in total. BERT has token classification by fine-tuning the model. It can be applied to tasks other than natural language processing. The model has two pre-trained tasks, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM model randomly masks a given sentence to 15% of the words in the input. The model predicts the masked words from the entire input masked sentences. During prediction, the NSP model concatenates two masked sentences as inputs. The model must then predict whether or not the two sentences followed each other. During preprocessing, the texts are lowercase to vocabulary size. A certain percentage of tokens in each sentence are randomly selected for masking. The default rate is 15% of the tokens. [MASK] replaces 80% of the masked tokens, random token replaces 10% of masked tokens, remaining 10% of masked tokens are left unchanged. Thus the embedding has special tokens called [CLS] at the beginning of each sentence, the token [SEP] to separate two sentences in a sequence and at the end of the sentence, and [MASK] to mask any word in the sentence. An overview of the BERT model for a classification task is shown in Figure 5.3. In the classification stage, the model classifies the given email as phishing or ham and also outputs the prediction accuracy.

5.3 EXPERIMENTAL RESOURCES AND DATASETS

In the proposed work, we used email as a primary resource. The required emails were obtained from two sources as discussed in chapter 2. Section 5.2 discusses dataset preparation procedures, and Table 5.1 lists prepared datasets with corpus sizes. Database-1 contains 3976 legitimate and 4216 phishing emails from the open source corpus, database-2 contains 12288 legitimate and 9391 phishing emails, and database-3 contains 12288 legitimate and 10639 open source phishing emails.

Table 5.1: Used datasets

Dataset	Ham emails	Phish emails	Total
Dataset-I	3976	4216	8192
Dataset-II	12288	9391	21679
Dataset-III	12288	10639	22927

5.4 EXPERIMENTAL RESULTS AND DISCUSSION

To assess the proposed model, we ran three different experiments with three different datasets, which are listed in Table 5.1. Table 5.2 summarizes the findings of all three experiments. The basic experimental setup required to carry out these experiments is provided below.

5.4.1 Basic experimental setup

To begin with basic experimental setup, the programming language used is *Python*, and the libraries used are *pandas*, *numpy*, *seaborn*, *nlk*, and *ktrans*. The tool used to develop model is Jupyter Notebook and the operating system used is Ubuntu-18.04.6 LTS. The hyper parameters used to develop proposed model are, The MODEL_NAME used is bert-base-uncased, MAXLEN size is set to 128, The Batch size used is 32, Learning rate is set to 5e-5, and Number of epochs used is 10. The data used for training is 75% of the total dataset size and testing of 25% of the dataset size for all three datasets and the SEED used is 2020 random state.

5.4.2 Results and discussion

In this section the experimental procedures are discussed. The model uses randomly selected data for training and testing with a ratio of 75:25% of the total dataset size. After removing unwanted characters and symbols, the data is feed to the BERT transformer model. The BERT base model has 12 transformer layers, 768 hidden sizes, and 12 self-attention heads. The transformer learns and selects parameters from the input data. For Dataset-I and II, the TFBertMainLayer is set to 109482240 parameters associated with dropout_37 and dense classifier. The models total parameters and trainable parameters for the Dataset-I is set to 109484547. The size and contents of Dataset-III varies, the model parameters and layers also vary with respect to the complexity of the data.

Table 5.2: Model performance with all three datasets

Dataset	Learning rate = 5e-5 (0.0337)				
	Training accuracy	Training loss	Validation accuracy	Validation loss	Training time (seconds)
Dataset-I	0.9995	0.0023	0.9951	0.0251	951
Dataset-II	0.9903	0.0253	0.9856	0.0609	2504
Dataset-III	0.9902	0.0260	0.9897	0.0254	2699

The objective of the proposed model is to classify the given input as either positive or negative (0 or 1) to indicate phishing or legitimate email. The model performance with dataset-I, dataset-II, and dataset-III are tabulated in Table 5.2. The model performance with dataset-I is 99.95% training accuracy, 99.51% validation accuracy, 0.0023 training loss, 0.0251 validation loss, and time taken to build the model is 951 seconds. The validation accuracy and validation loss graphs are shown in Figure 5.4. The results of all evaluation matrix for the dataset-I are tabulated in Table 5.3. According to Table 5.3, the precision is 99.20%, recall is 99.80%, and f-score is of 99.50%. The obtained results with open source datasets are competitive and outperformed all other existing works.

Using in-house data Dataset-II, the model performance is analyzed with a training accuracy of 99.03%, validation accuracy is 98.56%, training loss is of 0.0253, validation loss is of 0.254, and time taken to train and validate the model is 2504 seconds. All relevant metrics were measured and recorded, with the detailed results presented in Table 5.3. According to the results, the obtained precision is 99.74%, recall is 97.76%, and f-score is of 98.74%. The model accuracy charts for the given input is shown in Figure 5.5. The results of proposed model with Dataset-II proves that the prepared dataset is appropriate and suites best for the phishing classification.

Dataset-III is a combination of in-house legitimate and Nazario's phishing emails. The model performed equally well with the selected data and achieved training accuracy of 99.02%, validation accuracy of 98.97%, training loss is of 0.0260 and validation loss is of 0.0254. The accuracy and loss charts are shown in Figure 5.6, performance metrics are tabulated in Table 5.3.

5. Phishing Classification based on Text Content of an Email Body using Transformers

Table 5.3: Obtained results using transformers with all three datasets

Measure	Dataset-I	Dataset-II	Dataset-III
Sensitivity/Recall	0.9980	0.9776	0.9813
Specificity	0.9925	0.9965	1.0000
Precision	0.9920	0.9974	1.0000
Negative Prediction Value	0.9981	0.9702	0.9777
False Positive Rate	0.0075	0.0035	0.0000
False Discovery Rate	0.0080	0.0026	0.0000
False Negative Rate	0.0020	0.0224	0.0187
Accuracy	0.9951	0.9856	0.9897
F-score	0.9950	0.9874	0.9905
MCC	0.9902	0.9709	0.9795

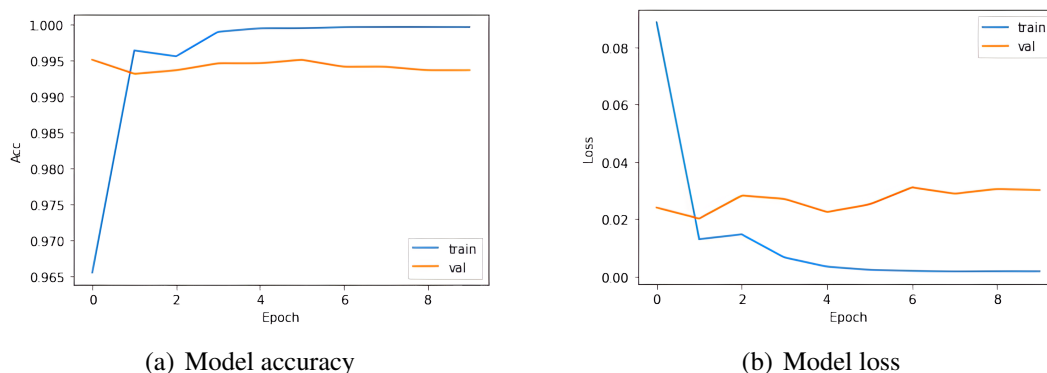


Figure 5.4: Accuracy and loss charts for Dataset-I

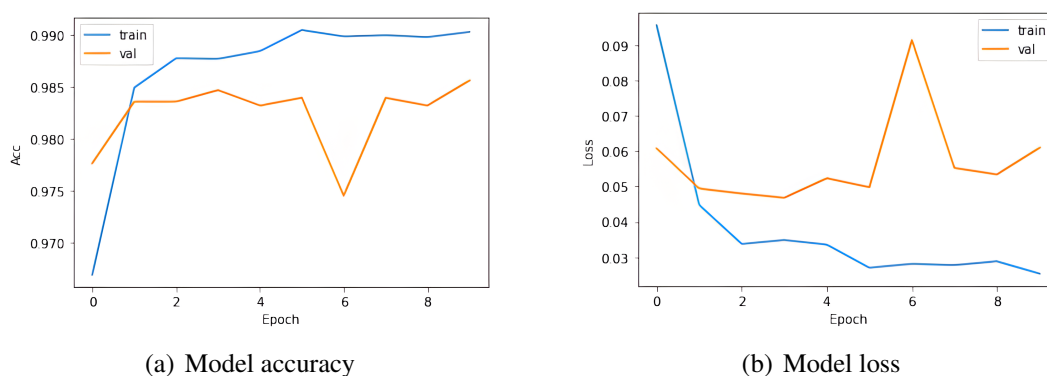


Figure 5.5: Accuracy and loss charts for Dataset-II

5.4.3 Result analysis

The primary input for the current study is the email body text content. The proposed model eliminates all unrelated and garbage content in the email body text and examines the content. Selected three datasets are processed independently with the proposed model. BERT base model with suitable parameters produces a competitive performance

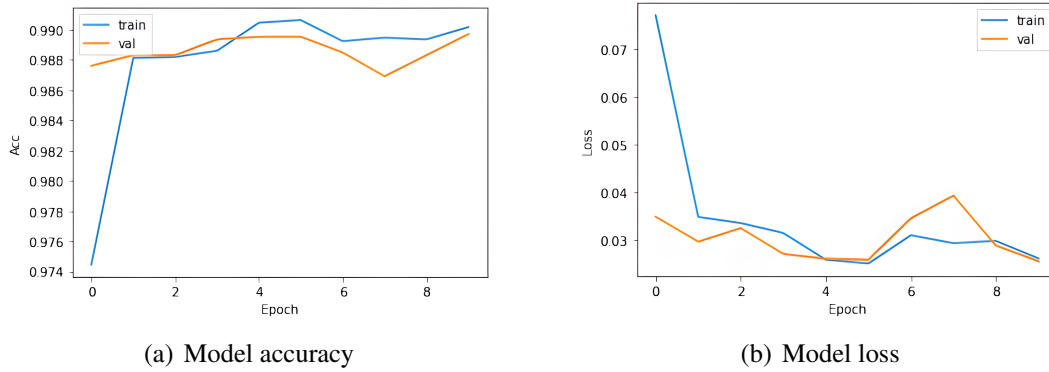


Figure 5.6: Accuracy and loss charts for Dataset-III

with all three datasets, and the results are tabulated in Tables 5.2 & 5.3 and shown in Figures 5.4, 5.5, and 5.6. Due to BERT’s pre-trained nature, the current method eliminates the most time-consuming feature selection and extraction processes. The proposed model suites for text processing and classification and achieves the highest accuracy using individual datasets for phishing classification.

5.4.4 Comparison study

In Table 5.4, we have provided a compilation of recent studies that exclusively utilize the text contents of email bodies as input for their models. The majority of these studies made use of commonly available open source datasets and achieved satisfactory outcomes. In our proposed work, Dataset-I is used, consisting of the Nazarios phishing corpus and SpamAssassin ham datasets, which were previously utilized by Ramanathan and Wechsler (2012). By utilizing Dataset-I, our model surpassed all other existing works, demonstrating an impressive accuracy of 99.51% during the 1st and 6th epochs out of a total of 10 epochs. Furthermore, the model achieved the highest precision, recall, and f-score values, reaching 99.20%, 99.80%, and 99.50% respectively. The proposed model also exhibited strong performance when applied to Dataset-II and Dataset-III, achieving accuracies of 98.56% and 98.97% respectively. The experimental results conclusively demonstrate that the proposed model outperforms other existing techniques.

5. Phishing Classification based on Text Content of an Email Body using Transformers

Table 5.4: Comparison Study

Author(s)	Datasets	Precision	Recall	F-score	Accuracy
Alhogail and Alsabih (2021)	CLAIR-ACL	0.985	0.983	0.985	0.982
Castillo et al. (2020)	Enron, APWG, and Private	-	-	-	0.9568
Bountakas et al. (2021)	Enron & Nazario	0.9863	0.9931	0.9897	0.9895
Bountakas et al. (2021)	Enron & Nazario	0.85	0.8409	0.8454	0.8449
Hiransha et al. (2018)	IWSPA-AP 2018	-	-	-	0.942
Ramanathan and Wechsler (2012)	SpamAssassin, Nazario's Phishing Corpus and Enron	0.997	0.997	0.997	0.977
Proposed work	Dataset-I	0.9920	0.9980	0.9950	0.9951
	Dataset-II	0.9974	0.9776	0.9874	0.9856
	Dataset-III	1.0	0.9813	0.9905	0.9897

5.5 SUMMARY

This chapter presented a novel phishing email classification model based on BERT transformers using only email body text. We also built an internal email dataset and validated it with our proposed model. For the open source data, the proposed model with dataset-I achieved the highest accuracy of 99.51%. Furthermore, the proposed work outperformed all other existing works using only the email body text feature for the identification or detection of phishing emails.

CHAPTER 6

EFFICIENT DEEP LEARNING TECHNIQUES FOR THE DETECTION OF PHISHING WEBSITES

Researchers have explored various methods to detect phishing websites, which are fraudulent websites designed to trick users into sharing sensitive information. One approach is heuristic analysis, which involves using a set of predefined rules to identify malicious websites. This method analyzes patterns and features such as domain names and URLs to detect suspicious sites. Another method is supervised machine learning, which uses labeled data to train a model to identify phishing sites with high accuracy. This approach involves feeding the model examples of both legitimate and phishing websites so that it can learn to differentiate between them. Unsupervised machine learning, on the other hand, uses unlabeled data to train a model to detect malicious websites. This method employs clustering and anomaly detection techniques to identify suspicious sites without prior knowledge of their classification. Deep learning is a subset of machine learning that uses neural networks to learn from large datasets. Deep learning can analyze the content and structure of websites as well as user behavior to detect phishing websites. The chapter proposes new models for detecting phishing URLs using deep neural networks, long short-term memory, and convolutional neural networks, using only 10 features from the work of Rao and Pais (2019). These models aim to improve the accuracy and efficiency of phishing website detection.

6.1 INTRODUCTION

The advent of the internet has brought about significant changes in various areas, including social networking, communication, banking, marketing, and service delivery. The number of users availing themselves of these internet services is increasing rapidly. However, as communication technology grows to meet human needs, adversaries also grow to disrupt communication and steal sensitive information by tricking users through malware or phishing websites. Phishing is a fraudulent technique used in the cyber world, where a phisher sends bait in the form of a replica of a legitimate website and waits for users to fall prey. The phisher succeeds when a user becomes a victim by trusting the fake website. Recent research scientists have been paying more attention to phishing attacks to prevent damage to innocent internet users. Several consortia, such as NSFOCUS, Anti-Phishing Working Group (APWG), and others, have conducted surveys of such attacks. These surveys aim to identify the extent of phishing attacks, the methods used by phishers, and the impact of such attacks on individuals and organizations. By understanding these aspects, countermeasures can be developed to protect internet users from falling prey to phishing attacks.

The Anti-Phishing Working Group (APWG) is a non-profit international consortium that analyzes phishing attacks reported by its members, which include security products, service-oriented organizations, law enforcement agencies, government agencies, trade associations, and regional international treaties and communications organizations such as BitDefender, Symantec, McAfee, VeriSign, and others. APWG publishes statistical reports on phishing trends across cyberspace periodically, either quarterly or half-yearly. According to the latest APWG (2018) report, there were 263,538 reported phishing attacks, representing a 46% increase compared to the fourth quarter of 2017.

Phishing attacks can take different forms, including email phishing, website phishing, and malware. In email phishing, attackers send spoofed emails pretending to be from trusted companies or organizations. In website phishing, phishers create websites that mimic real sites and advertise on other website contents or technology giants such as Facebook, Twitter, Google, etc. Some phishing sites use security indicators such as Hypertext Transfer Protocol Secure (HTTPS) and the green padlock, which can make

it challenging for users to differentiate between real and fake sites. Various techniques have been proposed to detect and prevent phishing attacks. These techniques include:

- *Listing-based detection:* The first technique for detecting and preventing phishing attacks is called listing-based detection. This technique is used by most web browsers, such as Chrome, Mozilla, and Opera, and involves maintaining a database of blocked and permitted URLs. The database of blocked URLs is called a blacklist, while the permitted URLs are stored in a whitelist. The browser compares the URL of a website the user is trying to access with the entries in the blacklist and whitelist databases. If the URL matches a blacklist entry, the website is blocked, and the user is alerted. However, if the website is not found in the whitelist database, even legitimate sites may be blocked. On the other hand, if the website is not in the blacklist database, it may be a phishing site, and the user may fall prey to the attack. This technique is not foolproof, as it may fail when encountering zero-day phishing sites that are newly created and not yet in the database. Moreover, phishers may slightly change the URL to bypass this technique. Therefore, it is crucial to update the list regularly to keep up with the increasing number of phishing attacks.
- *Heuristic-based detection:* Another technique for detecting and preventing phishing attacks is heuristic-based detection. This technique relies on extracting features from phishing sites and using them to identify potential attacks. The features may include the domain name, URL structure, content, images, and other elements of the website. By analyzing these features, the system can determine if the site is a phishing site or not. However, this technique has its limitations, as not all phishing sites will have the same heuristic features, which can reduce detection rates. Furthermore, this technique can be easily bypassed if the attacker knows which detection features the system is using and designs their phishing site to avoid those features.
- *Visual-similarity based detection:* Visual-similarity based detection is a technique used to detect phishing attacks by comparing the visual elements of a suspicious

website with a database of logos, screenshots, favicons, and Document Object Models (DOM) of legitimate websites (Fu et al. 2006; Hara et al. 2009; Rao and Ali 2015; Wenyin et al. 2005). If the similarity score is higher than a certain threshold, it is assumed that the suspicious site has mimicked some legitimate sites and is declared as phishing. However, this technique has limitations as phishers could easily bypass this security system by making slight changes to visual elements without changing the contents of the website. Therefore, visual-similarity based detection should be used in conjunction with other detection methods to improve accuracy and reduce false positives.

- *Conventional Machine-learning based detection:* Heuristic detection has a limitation that it cannot adjust to changes in phishing sites, even minor ones, resulting in missed detections. To overcome this problem, machine learning techniques have been applied to provide flexibility to the heuristic model. This approach involves training a machine learning model with datasets that contain values of features extracted using a heuristic approach. Various algorithms such as Support Vector Machine Decision Tree (SVMDT), Random Forest (RF), Sequential Minimum Optimization (SMO), Principal Component Analysis Random Forest, J48 tree, Multilayer Perceptron, among others, are used for this purpose. These algorithms can detect zero-day phishing attacks when trained with heuristic model features. With a vast training dataset, these algorithms perform better as they can learn most of the possible variations that phishing sites may have. According to Rao and Pais (2019), the accuracy achieved using machine learning techniques for detecting phishing sites is about 99.5%. In a survey conducted by Khonji et al. (2013), the detection of phishing sites with an accuracy of over 99% was possible using machine learning techniques. The performance of the machine learning algorithm depends on the size of the training data, the quality of the extracted features, and the values of certain hyperparameters used to optimize accuracy.
- *Deep learning based detection:* Deep learning is a machine learning technique that learns features directly from data. The data may be images, text, or sound. Deep learning requires a large amount of labeled data and makes it possible for

the Graphical Processing Unit (GPU) to train deep networks in less time. New trends have been made to exploit Deep Neural Network (DNN) techniques such as multi-layer feed-forward network (Zhang and Yuan 2012), Convolutional Neural Networks (CNN) (Le et al. 2018) and Recurrent Neural Network (RNN) (Bahnsen et al. 2017) to detect and prevent phishing attacks. These networks are trained through multi-featured data sets obtained using heuristic methods. Bahnsen et al. (2017) trained the RNN over the URL character sequence. They argued that each character sequence has correlations, i.e., nearby characters in the URL are likely to be connected. These sequential patterns are important because they can be used to improve predictor performance. Le et al. (2018) used CNN to learn sequential URL behavior. They adopted two techniques that are CNN character level and CNN word level, which identify unique characters and words. Each character or word is represented as vector and trains the vectors over CNN to learn the sequential behavior of the URL to identify the phishing URLs.

The use of machine learning algorithms that are trained on data sets containing heuristic methods has resulted in the development of numerous approaches to combat phishing sites. Many studies (Huh and Kim 2011; Jain and Gupta 2018c; Khonji et al. 2013; Whittaker et al. 2010) have utilized external sources such as Google or Bing search results, Alexa¹ ranking, and WHOIS² to identify phishing sites. However, some phishing sites hosted on compromised domains can evade such techniques. According to the report by APWG (2014), while most phishing sites do not survive for more than a day, those hosted on compromised sites can persist for more extended periods, highlighting the limitations of the existing methods and contributing to the increase in phishing attacks. Hence, there is a need for a more accurate mechanism that can prevent phishing attacks while minimizing the use of third-party services with fewer features. The heuristic method captures potent and specific features that are robust enough to detect even zero-day phishing attacks. These methods were employed to extract the necessary features for training our multi-layer DNN, Long Short-Term Memory (LSTM) Network, and CNN. We also endeavored to optimize the hyperparameters of these net-

¹<https://www.alexacom/topsites>

²<https://www.whois.com>

works to achieve the best possible accuracy with minimal features.

In a study conducted by Rao and Pais (2019), they employed Random Forest (RF) and its variations as classifiers, along with a comprehensive set of features, to classify phishing sites. In our current study, we have utilized deep learning algorithms such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Deep Neural Network (DNN) to detect phishing websites. Additionally, we used an information gain algorithm to select the top-performing features among our proposed features and utilized them for classifying phishing websites. The feature selection process resulted in a reduction of the number of features from 18 to 10 while achieving the same accuracy as that of our previous work. Among these ten features, six are features proposed by other researchers, while four are features proposed in our earlier work. Furthermore, our approach relies less on third-party services than the previous work.

The related works with deep learning classifiers are summarized in Table 6.1. The table gives a comparison between the proposed method and all other approaches using six different metrics. These metrics include the detection of phishing sites that replace textual content with an image (Image-based phishing), detection of phishing sites that contains most of the hyperlinks directed towards a common page (Common Page Detection), detection of phishing sites that are hosted in any language (language independence), detection of phishing sites that consists of maximum number of broken links (Broken links), detection of phishing sites based on different models, and the number of features used for classification of phishing sites.

The current chapter makes the following research contributions,

1. An innovative Information Gain (IG) algorithm is introduced to effectively identify the most effective features for detecting phishing URLs.
2. Novel DNN, LSTM, and CNN models are proposed specifically designed for phishing URL detection, utilizing a minimal set of 10 features.
3. The performance evaluation of these models reveals promising accuracies of 99.52%, 99.57%, and 99.43% for DNN, LSTM, and CNN respectively.

Table 6.1: Summary of related work in comparison with proposed work

Techniques	Image based Phishing	Common page based Phishing	Language independence	Broken Links	Models	Features.
Rao and Pais (2019)	Yes	Yes	Yes	Yes	Machine Learning	18
Guarda et al. (2019)	No	No	No	No	SMO	15
Yao et al. (2018)	No	No	Yes	No	Neural networks	17
Parsons et al. (2019)	Yes	Yes	Yes	No	PNN K-Modoids Clustering	30
Al-Musib et al. (2021)	No	No	Yes	No	Gated Recurrent Neural Network	Direct URLs
Balim and Gunal (2019)	No	No	Yes	No	Convolution Neural Network	Direct URLs
Jia et al. (2021)	No	No	Yes	No	Recurrent Neural Network	Direct URLs
56	No	No	Yes	No	CNN-LSTM Hybrid Network	Direct URLs
Mondal et al. (2022)	Yes	Yes	Yes	No	Neural Network	30
Sonowal (2022a)	Yes	No	Yes	Yes	Deep DBN	8
Proposed Model-I	Yes	Yes	Yes	Yes	Deep DNN	10
Proposed Model-II	Yes	Yes	Yes	Yes	Deep LSTM	10
Proposed Model-III	Yes	Yes	Yes	Yes	Deep CNN	10

6.2 DEEP LEARNING BASED URL CLASSIFICATION MODEL

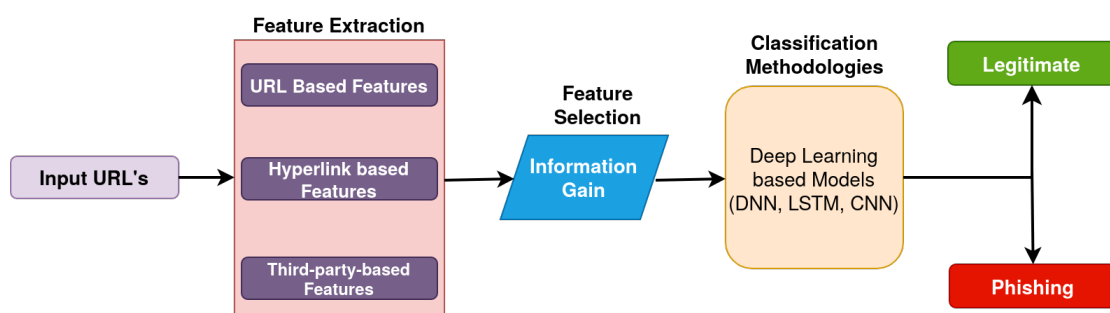


Figure 6.1: Architecture of Proposed Model

The aim of this research is to identify the legitimacy of a given URL using minimal,

distinctive features with the help of deep learning classifiers. The proposed system's architecture is presented in Figure 6.1, which comprises feature extraction, feature selection, and classification methods. Initially, a set of webpage URLs is fed as input to the feature extractor, which extracts necessary features from three sources, including URL obfuscation, hyperlink, and third-party based. The extracted features are then forwarded to the Information Gain (IG) feature ranking algorithm, which helps to select the most effective features by carefully analyzing their dependencies. The top 10 best-performing features are then trained through various deep learning techniques to produce the URL's status as either legitimate or phishing. The individual models' comprehensive explanations are provided below.

6.2.1 Feature Extraction

The features are extracted from three sources such as,

- URL Obfuscation features
- Hyperlink based features
- Third-Party based features

These features are extracted using Selenium with Python language, an HTML parser, and BeautifulSoup for parsing the websites. The selection of prominent features from the extracted features is carried out using the information gain mechanism. The information gain for the features proposed by Rao and Pais (2019) is given in Table 6.2.

URL Obfuscation Features:

These are the properties that can be obtained from the URL itself, without requiring access to the website's content or relying on third-party services. To define various URL-based features, we first need to comprehend the anatomy of a typical URL. A URL is a unique Uniform Resource Identifier (URI) that is utilized to locate existing resources on the internet. When a web client requests the server for resources such as HTML, CSS, images, videos, or other hypermedia, a URL is used. A URL typically comprises of four or five components. The typical structure of URL is, (`http:`

`//www.reg.signin.nitk.com.pk/secure/login/web/index.php`). It consists of the following parts.

- **Scheme:** The scheme is used to identify the used protocols, Hypertext Transfer Protocol (HTTP) or HTTP with Secure Sockets Layer (HTTPS).
- **Hostname:** The hostname identifies the machine that contains resources. The hostname includes the Generic top-level domain (gTLD) and Country-code top-level domain (ccTLD). In the given example **reg.signin** indicates subdomain, **nitk** is primary domain, **com** is gTLD and **pk** is ccTLD.
- **Path:** The path identifies the basic or required information in the host that the web client wants to access. From the given URL example, the pathname is **secure/login/web/index.php**.
- **A query String:** When a query string is used, the path component follows and provides a string of information that the resource can use for some purpose. The query string is usually the name and value pair string. Name and value pairs are separated by an ampersand (&). For example: In the URL `http://www.google.co.uk/search?q=url&ie=utf-8,?q = url&ie = utf - 8` is the query string with name and value pairs as $q = url$ and $ie = utf - 8$.

In Rao and Pais (2019), the authors proposed five URL obfuscation features (UF1, UF2, UF3, UF4, UF5). Out of those, two features are poorly performing when applied to the IG algorithm shown in Table 6.2. The best performing three features have been selected, those are,

1. UF1: Dots in Hostname
2. UF3: Length of URL
3. UF5: Presence of HTTPS

Hyperlink based features:

These features are extracted from the hyperlinks in the source code of a website. The

Table 6.2: Information gain of individual features

Features from Rao and Pais (2019)	Information gain
UF1 - Dots in Hostname	0.0874
UF2 - URL with @ symbol	0.00797
UF3 - Length of URL	0.28293
UF4 - Presence of IP	0.00523
UF5 - Presence of HTTPS	0.07321
TF1 - Age of Domain	0.29139
TF2 - Page Rank	0.88344
TF31 - Website in search engine Results-title	0.15664
TF32 - Website in search engine Results-copyright	0.16603
TF33 - Website in search engine Results-description	0.27909
HF1 - Frequency of domain in anchor links	0.21588
HF2 - Frequency of domain in CSS links,image links and script links.	0.04654
HF3 - Common page detection ratio in website	0.40058
HF4 - Common page detection ratio in footer	0.29128
HF5 - Null links ratio in website	0.25015
HF6 - Null links ration in footer	0.08162
HF7 - Presence of anchor links in Website	0.14237
HF8 - Broken links ratio	0.20216

hyperlink is an electronic document element that connects from one source to another. The web source may be an image, program, HTML document or HTML document element. Rao and Pais (2019) technique consists of 8 hyper link based features that are used for phishing detection. We have selected six best performing features among eight features based on the results of information gain analysis shown in Table 6.2, and the selected features are given below:

1. HF1: Presence of domain in anchor links
2. HF2: Frequency of css links, image links and script links
3. HF3: Common page detection ratio in website
4. HF4: Common page detection ratio in footer
5. HF7: Presence of anchor links in website
6. HF8: Broken links ratio

It may be observed that HF5 and HF6 have performed better in information gain but

they have been eliminated since these features characteristic are captured by HF3 and HF4 (Srinivasa Rao and Pais 2017).

Third-Party based feature:

In this section, we use third-party services such as WHOIS, Alexa and Search engine for the extraction of third-party based features. Surprisingly, out of these three features, the Alexa rank based feature performed significantly better in the information gain. Even though other third-party features performed better than the other (URL obfuscation, Hyperlink based) features, we have not considered them in our feature selection to reduce the dependency on third-party services.

TF2: Alexa Ranking is a third-party based service used to classify the phishing sites. The rationale behind this feature is that phishing sites are low ranked and target websites are highly ranked. This feature checks the rank of a suspicious website in the Alexa database. To calculate the rank, an HTTP request is sent to (`http://data.alexacom/data?cli=10&url="+domain`) and use an XML parser to get the Alexa ranking.

$$Pagerank = \begin{cases} 0 & \text{if rank is not found} \\ rank & \text{Otherwise} \end{cases}$$

The selected features from the above three sources are highlighted and marked as selected features in Table 6.3.

6.2.2 Feature Selection

We have used information gain as a ranking criterion to score the features and by applying threshold, we have filtered out prominent features. The intuition behind ranking is to evaluate the relevance of the features for the detection of phishing websites. And the relevance of feature implies that each feature may be mutually exclusive to each other, but it must not be completely independent of class labels. There must exist a relation between feature and class labels. And the features which are irrelevant and have no relation or abysmal role can be discarded. Hence, our primary aim behind using this technique is to rank features on the basis of its relevance and influence on the class labels and thus could be used in the feature reduction process. Information gain (Quinlan

Table 6.3: Selected Features

Features from Rao and Pais (2019)	Selected Features (✓)
UF1	✓
UF2	-
UF3	✓
UF4	-
UF5	✓
TF1	-
TF2	✓
TF31	-
TF32	-
TF33	-
HF1	✓
HF2	✓
HF3	✓
HF4	✓
HF5	-
HF6	-
HF7	✓
HF8	✓

1986) is measured based on the entropy of a system, which is defined as a degree of disorder and impurity in the system. And information gain is defined as a reduction in the impurity and bringing more certainty in the system. And for feature ranking purposes, we have calculated information gain on the entire dataset. Summarizing information gain looks at each feature in isolation and is calculated on each feature independently. By computing information gain of each feature independently, we get a quantitative measure of significance and relevance of this feature on class labels. Computation of information gain for a feature involves two steps:

1. Compute entropy of the class label for the entire dataset. It can be computed by the formula:

$$Info(D) = \sum_{i=1}^m p_i * \log_2 p_i \quad (6.1)$$

where $m= 2$ i.e., the total unique number of class labels (phishing, legitimate) and in our dataset, it is two. D represents a feature of a dataset. Hence, each feature has some instances belonging to one class and remaining to another class. p_i represents the probability of instances of D that belongs to i^{th} class. We can

compute probability p_i by counting the number of instances of D that belongs to i^{th} class then we divide by the total number of instances of D. Once we get p_i for all i, we use equation 6.1 to calculate entropy of D.

2. Computation of conditional entropy for each unique value of that feature:

The calculation of conditional entropy requires a frequency count of the class label by feature value. The feature value can be continuous as well as discrete.

i. For discrete-valued features, it can be calculated by the formula:

$$Info_A(D) = \sum_{i=1}^v |D_i| / |D| * Info(D_i) \quad (6.2)$$

Where v is equal to total unique discrete values present in the feature value, D_i represents a count of i^{th} type of value in feature, and D is the total count of feature value.

ii. For continuous-valued features, we have sorted feature value and have divided them in \sqrt{n} bins, where n is equal to the total count of feature values. Now we have n different classes and it can be treated as discrete-valued features, and the equation 6.2 is used to calculate the conditional entropy of continuous feature.

Now the information gain is calculated using the equation given below:

$$InformationGain(A) = Info(D) - Info_A(D) \quad (6.3)$$

The information gain for the features proposed by Rao and Pais (2019) is given in Table 6.2.

6.3 IMPLEMENTAION

Given a list of website URLs, we have trained and cross validated a proposed deep learning based model to identify as legitimate URL or phishing URL. We have used the Selenium library in Python and Firefox web driver to get screenshots of website URLs, and also to download the source code. We used BeautifulSoup in Python to parse the source code to extract the required features. Screenshots and status codes are further used to verify that contents have not changed while extracting features from source code. Extracted datasets are further examined manually and removed duplicates, legit-

imate URLs, and unwanted URLs (neither phishing nor legitimate) from the phishtank dataset. This process is to avoid legitimate sites that are treated as phishing and reduce the processing time by avoiding unwanted comparisons.

6.3.1 Tools Used

We have implemented python scripts to extract all features using Python 3.6 from URL and URL content. We collected phishing URLs from the PhishTank³ website, and legitimate sites from the Alexa databases. When these URLs are fed as inputs to the python script, all the required features are extracted and stored in text files. These extracted features are transferred to deep learning algorithms to train and cross validate so that it can start classifying URLs into legitimate and phishing sites. We have implemented a deep learning algorithm with a TensorFlow package, an opensource machine learning framework implemented on top of python which supports parallel computing.

6.3.2 Datasets Used

We have used the dataset of Rao and Pais (2019) for all our experiments in this chapter. The dataset consists of 3526 instances out of which 2119 are phishing sites collected from PhishTank and 1407 legitimate sites collected from the Alexa database. These were further divided into categories of training sets and testing sets of 75% and 25% respectively for model evaluation.

6.3.3 Deep Learning Algorithms

To evaluate the performance of the feature set, the feature set have been trained and cross-validated against many different parameter combinations. In the multi feed-forward network, we must gather data based on feature sets and then tune the parameters to achieve maximum accuracy in phishing site classification. It is an essential process in which parameters must be set by training networks and validated across appropriate values. After achieving the right value, phishing sites can easily be classified with the highest probability. We used Python programming language along with the TensorFlow library to implement deep learning algorithms. From various combinations of hidden

³<http://www.phishtank.com/index.php>

layers, we found that the deep neural network with five hidden layers achieved the best results. And this can be understood that the features we have extracted in the nonlinear, separable, and complex functions need to be represented most effectively. The proposed deep feed-forward neural network comprises of 7 layers, where five layers are hidden, one input layer and one output layer. All layers were followed and standardized by the Rectified Linear Unit (ReLU) or Sigmoid function. The first four layers were followed by the ReLU function and the output layer using the sigmoid function. The rationale behind batch normalization is that it speeds up training by reducing the internal covariate shift and reducing overfitting. ReLU activation has replaced Sigmoidal or Tanh activation functions in hidden layers due to their tendency to learn faster than sigmoidal or tanh, avoiding significant delays in the rate of gradient descent convergence after an initial set of iterations. In the current research we have used three deep learning algorithms to classify the websites as phishing or legitimate those are DNN, LSTM, and CNN. The formal description of DNN and LSTM are discussed in Chapter 4 section 4.2.1, the formal description of CNN is discussed below.

Formal Description of CNN : Convolutional Neural network is similar to an ordinary deep neural network. These networks consist of neurons that have weights and bias, which are updated and made to learn. Each of these neurons receives inputs that are converted into a linear combination of dot products of weights and input bias. But instead of fully connected hidden layers, it performs convolution on input layers $x \in R^{L*B}$. Convolution is performed using convolution operator \otimes of length L with stride s, and consist of convolving filter $W \in R^{B*K}$.

Generally, convolutional neural networks are used with images due to the high correlation between pixels and networks. CNN's can figure out relations and different features using convolutional techniques, which are used in conjunction with the pooling layer, and batch normalization is done before passing it to any activation function (Krizhevsky et al. 2017; Le et al. 2018). It has also been used in Natural Language Processing after character encoding due to the correlation between character sequences (Le et al. 2018; Pham et al. 2016).

In this work, the selected ten features from the Information Gain algorithm are fed to

the CNN model to identify the status of the suspicious site. The proposed CNN model consists of eight layers (six convolution and two dense layers). In the first layer, the input is passed to the convolution layer, and the output of this layer is activated using tanh function (Eq. 6.4). Then the activated output is subjected to batch normalization and pooling. The obtained output is passed to the next convolutional layer. In this way, we have six convolutional layers connected sequentially in which the output of one layer is the input of the next. At the seventh layer we densed the output of sixth convolutional layer to 500, and again activated using tanh function, then at the end densed it to 1. The output of the tanh function is passed to sigmoid activation function (4.12) for output in range of (0,1). And then the loss function (4.13) is calculated and being optimized using Adam Optimizer. Then we have a backpropagation method where variables are updated, and thus network learns.

$$\tanh = (e^{2x}-1)/(e^{2x}+1) \quad (6.4)$$

We have used ten features extracted in the feature selection process. The size of our dataset is 3526. So we converted our dataset into the dimensionality of (3526, 10, 1) and passed it to our CNN model for phishing detection.

6.4 RESULTS AND DISCUSSIONS

We conducted experiments to evaluate the performance of our DNN, LSTM, and CNN models with different features and parameters. All experiments were conducted with the same dataset of 3526 instances. Each experiment has been repeated, and data has been randomly selected from the dataset. For evaluating our model, we have used accuracy (Eq. 6.5) and error (Eq. 6.6) rates as the main evaluation metrics. To calculate the same, we considered the phishing sites as condition positive (P), where P represents the total number of phishing sites in our dataset. The legitimate sites are termed as condition negative (N), where N represents the total number of legitimate sites in our dataset. The correctly classified phishing sites are termed as True Positive (TP), which is calculated as the ratio of correctly classified phishing sites out of the total number of phishing sites (P). Correctly classified legitimate sites are termed as True Negative (TN), which is calculated as the ratio of correctly identified legitimate sites out of the total number of legitimate sites (N).

- **Accuracy (ACC)** : Measure the legitimacy and phishing rate of the total number of websites.

$$ACC = \frac{TP + TN}{P + N} \quad (6.5)$$

- **Error Rate (ERR)** : Measure the rate of legitimacy or phishing from incorrectly classified websites.

$$ERR = 1 - \frac{TP + TN}{P + N} \quad (6.6)$$

6.4.1 Validation of Selected Features using DNN

Our work of feature retention and rejection based on inferences that we have drawn from information gain listed in Table 6.2. We have retained those features which have higher information gain. In this section, we are validating our claim of feature selection using DNN.

To validate our selection of features, we have conducted three experiments. The First experiment is conducted by supplying all 18 features of the work Rao and Pais (2019) to DNN. The overall accuracy obtained using 18 features is 97.95%. The results of the individual accuracy is tabulated in Table 6.4. The testing accuracy of individual feature varies from 60.86% (UF2, UF4) to 96.47% (TF2). Based on the accuracy, we have eliminated two URL based features whose testing accuracy is less than 61% (UF2, UF4). We have eliminated two hyperlink based features HF5, HF6 since their functionalities are taken care of by HF3 and HF4, respectively. And also, individual accuracy of HF5 and HF6 are less than HF3 and HF4. Experiment 2 is conducted after eliminating four features (UF2, UF4, HF5, HF6) from the total set of 18 features. The accuracy chart of training and testing using these 14 features is given in Figure 6.3. The overall accuracy obtained using 14 features is 99.20%. Experiment 3 conducted to evaluate features by minimizing third-party based features. The overall accuracy after eliminating four third-party feature is 98.97%.

The 10 features used in experiment 3 are the same which have the highest information gain as given in Table 6.2. The experimental results are in line with the information gain and they validate our selection of features.

Experiment 1: Evaluation of individual heuristic features using deep neural network: In this experiment, the performance of each individual feature has been evaluated and is given in Table 6.4. This has been done in order to know the individual contribution of each feature in determining accuracy. The features, which have higher accuracy in detecting phishing sites, have more relevance to class labels. And this relevance will be an experimental manifestation of our feature ranking process using IG algorithm. It will also be an experimental justification of retention and rejection of features using information gain. The overall accuracy obtained using 18 features is 97.95%. The accuracy chart using 18 features is shown in Figure 6.2. The obtained accuracy is less than the work proposed by Rao and Pais (2019) with the same set of features.

Experiment 2: Evaluation of model using 14 features: In this experiment, we have evaluated our model accuracy using fourteen features. The feature that we left out are UF2, UF4, HF5, HF6. We have summarized reasons for leaving out above mentioned features as follows:

- UF2 and UF4: The first two features have individual accuracy of less than 61%, which is the lowest in Table 6.4. In our information gain Table 6.2, two of the lowest value are 0.00797 and 0.00523 for UF2 and UF4, respectively. The lowest accuracy among other features and lowest information gain among other features, validate our claim of rejection in the feature selection process. These two features are the least relevance to class labels and their individual contributions are the lowest. Hence, our experimental analysis upholds the rejection of these features in order to reduce inhibition offered by the least performing network.
- HF3 vs HF5: HF3 is the ratio of the most common link to the total number of links in the URL web page, and HF5 is a ratio of null links to the total number of links in the URL web page. If the null link is the most common link, then both are equivalent, and if null links are less or absent, then null link ratio features are of no use. So, HF3 contains HF5 (Srinivasa Rao and Pais 2017). For removing features, we had considered the information gain of each feature. In Table 6.4,

Table 6.4: Accuracy of individual features

Features	Training Accuracy	Testing Accuracy
UF1	64.94	63.94
UF2	59.90	60.86
UF3	78.03	76.79
UF4	59.90	60.86
UF5	66.94	67.12
TF1	77.34	79.41
TF2	95.83	96.47
TF31	81.96	80.01
TF32	75.57	74.18
TF33	64.78	62.0
HF1	72.04	68.25
HF2	64.48	61.32
HF3	82.37	80.31
HF4	80.33	79.18
HF5	75.04	73.95
HF6	63.39	63.25
HF7	68.75	68.03
HF8	76.96	75.88

HF3 individual accuracy is higher than that of HF5. Hence, our retention of HF3 is experimentally justified.

- HF4 vs HF6: HF4 is the ratio of the most common link to the total number of links in the footer, and HF6 is the ratio of the null link to the total number of links in the footer. Both will have the same value if the most common link is a null link. And if null links are less or absent, it is shallow. HF3, therefore, contains HF5 (Srinivasa Rao and Pais 2017). In our feature selection process, we have considered higher information gain of HF4 over HF6. In our experimental analysis, the individual accuracy of HF4 is higher than that of HF6, which validates our claim.

After removing these features, accuracy has increased to 99.20%, and the same can be observed in Figure 6.3.

Experiment 3: Evaluation of features by minimizing third-party based features: In this experiment, we began with retaining all third-party features (TF1, TF2, TF31, TF32, TF33) shown in Table 6.3. The extraction of third-party features from

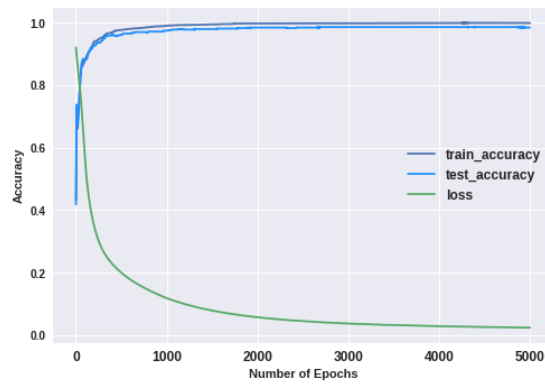


Figure 6.2: Network performance using 18 features

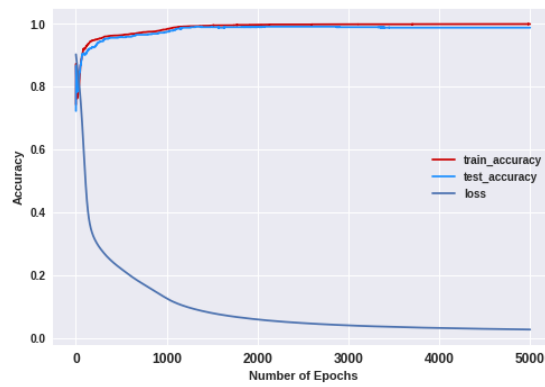


Figure 6.3: Accuracy chart with 14 features

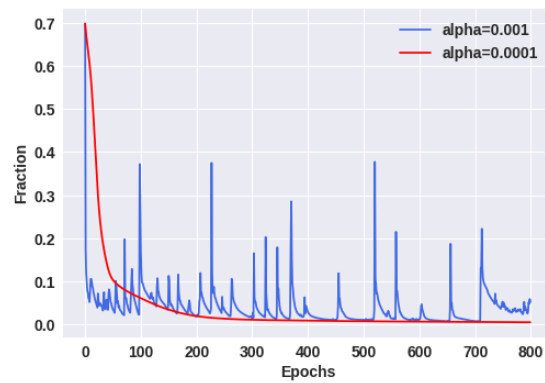


Figure 6.4: Learning rate with $\alpha=0.001$ and $\alpha=0.0001$

URL is a time consuming process. Hence, phishing URL detection can not be done in a faster time-bound manner. If any of these third-party features are not available, then we have to deal with missing data, and the accuracy of the model will drop. Hence, we removed all the third-party features including TF2, which has the highest information gain to test the robustness of our model. The accuracy of our model dropped to 90% after removing all the third-party features. Again we experimented with the inclusion of one third-party feature (TF2) and obtained an accuracy of 98.97% with 5000 epochs.

6.4.2 Results with DNN

In section 6.4.1, we have conducted three experiments using DNN to validate our features by comparing results obtained from the IG ranking algorithm. We have selected ten best performing features from experiment 3. The individual accuracies of the best ten features are shown in Figure 6.5. Experiment four is conducted using DNN on the dataset of Rao and Pais (2019) with selected ten features. The hyperparameters tuning is performed to optimize the model by selecting the learning rate (α), optimizer, number of hidden layers, number of nodes in layers, and number of epochs.

Experiment 4: Evaluation of model by tuning parameters: In experiment 3, we have not fine tuned the parameters to optimize our model. In experiment 4, fine tuning of parameters was performed to optimize our DNN model. The parameter fine tuning process is as follows:

- **Learning rate:** We began with $\alpha = 0.001$, keeping rest of the features as specified in Table 6.5. At this α value, our model's loss function converged rapidly, as indicated in Figure 6.4 with higher losses. In this case, we got 99.69% training accuracy and 98.97% testing accuracy. Hence, we increased α to 0.0001 and the loss function of our model began to converge as we can see in Figure 6.4 at about 800 epochs with lowest loss (error) of about 0.012%. We achieved a test accuracy of 99.20% and training accuracy of 99.51% with 5000 epochs. We further increased the number of epochs to 6000 by keeping α to 0.0001 and achieved consistent training accuracy of 99.55% and testing accuracy of 99.52% as shown in Figure 6.7 and all experimental results are tabulated in Table 6.6. We also, in-

6. Efficient deep learning techniques for the detection of phishing websites

creased the α to 0.00001 and the loss function converged at much higher epochs with the same accuracy. We, therefore, concluded that further decreasing of alpha would take more processing time without increasing model accuracy.

Table 6.5: Parameters for DNN

Layers	Number of units in layers	Learning rate	Optimizer	Epochs	Activation function
6	10, 19, 100, 200, 300, 1	0.0001	Adam Optimizer	6000	ReLU

Table 6.6: DNN Experimental Results

Experiment	α value	No. of Features	Epochs	Training Accuracy	Testing Accuracy
1	0.001	18	5000	98.71	97.95
2	0.001	14	5000	99.96	99.20
3	0.001	10	5000	99.69	98.97
4	0.0001	10	5000	99.51	99.20
	0.0001	10	6000	99.55	99.52

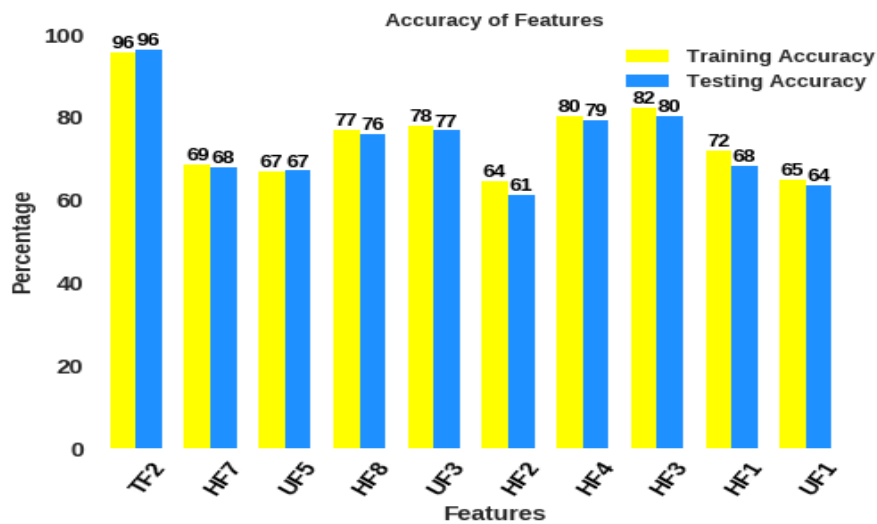


Figure 6.5: DNN Individual feature accuracy

- **Optimizer:** We used Adam optimizer to give us training accuracy of around 99.55% and test accuracy of 99.52%. We also tested our model with a Gradient descent optimizer, which makes our model very slow and less accurate. This can be clearly seen with the graph shown in Figure 6.6.

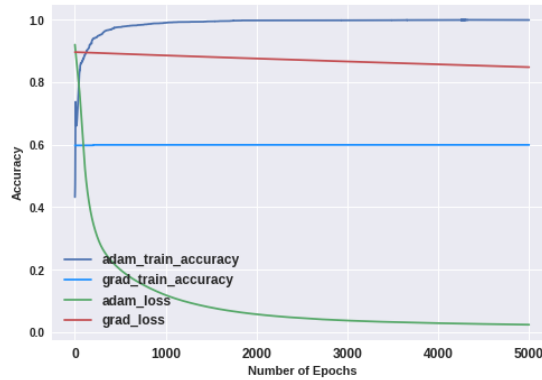


Figure 6.6: Comparison between Optimizers

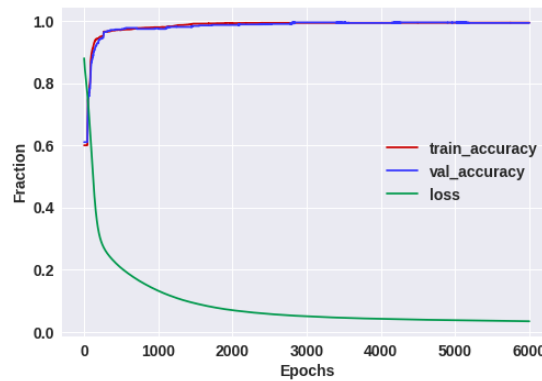


Figure 6.7: DNN accuracy with ten features

- Number of epochs:** We have used the iterative process to determine the total number of epochs to be used for the best performance of our model. We started with 500 epochs and increased by 500 until we got the minimum loss. The minimum loss means that if we keep iterating the model, the loss will continue to decline to some minimum value and then start fluctuating, so we have to stop at that minimum point. And it varies with different learning rates with different optimizers.
- Number of hidden layers:** The increasing number of hidden layers will result in increased network complexities because we will fit our data with the number of hidden layers and the number of hidden units. We have initialized with one hidden layer and moved progressively to identify the optimal hidden layers. Based on the empirical analysis, we achieved the model optimal results with four hidden layers. On further increasing the layers, the model showed non-promising results

with additional processing time.

- **Number of units in hidden layers:** There is no way to determine the number of hidden units in each layer. So we began with a smaller number of hidden units in hidden layers, as if they were faster but could not learn properly, and it resulted in less accuracy. So we increased and tested the accuracy of each of these configurations. It is observed that having more units slows down and leads to data over-fitting and therefore obtained lower accuracy.

Therefore, after all these experiments, we found that DNN is consistent by achieving 99.55% training accuracy and 99.52% testing accuracy with finalized 10 features. The individual features training and testing accuracies of DNN are shown in Figure 6.5. And the accuracy graph of selected features using DNN is shown in Figure 6.7. Experimental results are tabulated in Table 6.6. Due to the close difference between training and test accuracy, overfitting is reduced.

6.4.3 Results with LSTM

To check the effectiveness of our 10 features we conducted an experiment using LSTM. The parameters used for the experiment are given in Table 6.7. In LSTM, hyperparameters are tuned to achieve better accuracy. Those hyperparameters are used in experiment five to achieve promising accuracy.

Experiment 5: Evaluation of LSTM model by tuning parameters:

- **Number of LSTM Units:** We have started with 32 LSTM units. However, more LSTM units slow the processing of training. The network testing accuracy was much lower than that of training accuracy, which implies overfitting. After dropouts were used to reduce overfitting, there was a trade-off between accuracy and overfitting, and overfitting was reduced by reducing the precision of training. So we started decreasing LSTM units. And the above trend was seen until the number of LSTM units was reduced to four.
- **Learning rate (α):** This is one of the most important parameters to determine our model's convergence. If α kept large (near to one), the minimum convergence

point in the model contour can be skipped. If α kept small (near to zero), it will take a long time to reach the minimum convergence point. We began with $\alpha=0.0001$. But due to the lower magnitude of α , the network was slowly converging even after 5000 epochs and there was no improvement in accuracy with an increasing number of epochs and it was 98%. So we increased to 0.001. And at this α , we got our maximum accuracy (99.57%). But we continued till 0.1 (larger α) and it proved to be pointless as the network started oscillating and accuracy was decreasing.

- **Optimizer:** We have experimented with a simple gradient optimizer and applied an incremental approach by adding epochs. The experiment conducted with a maximum of 10000 epochs and observed the convergence rate was going slow with 10000 epochs. In this process, we observed the best possible accuracy with 700 epochs using Adam optimizer.

After tuning above mentioned parameters, we have achieved a training accuracy of 98.86% and testing accuracy of 99.57%. The accuracy graph that we have obtained is shown in Figure 6.9. The individual features testing and training accuracies are given in Figure 6.8. This figure shows that the training and testing accuracy of individual features are very close and prevents overfitting.

Table 6.7: Parameters for LSTM

Number of LSTM units	Learning rate	Optimizer	Epochs
4	0.001	Adam Optimizer	700

6.4.4 Results with CNN

The experiments are carried out using the CNN model to validate the performance of ten selected features. Table 6.8. lists the parameters used to conduct the experiments.

Experiment 6: Evaluation of CNN model by tuning parameters:

- **Window Size:** This is the size of a one-dimensional window that must be convolved sequentially. Because there are only 10 features in the dataset, the window size options are limited. So we started with a larger window size (7) and

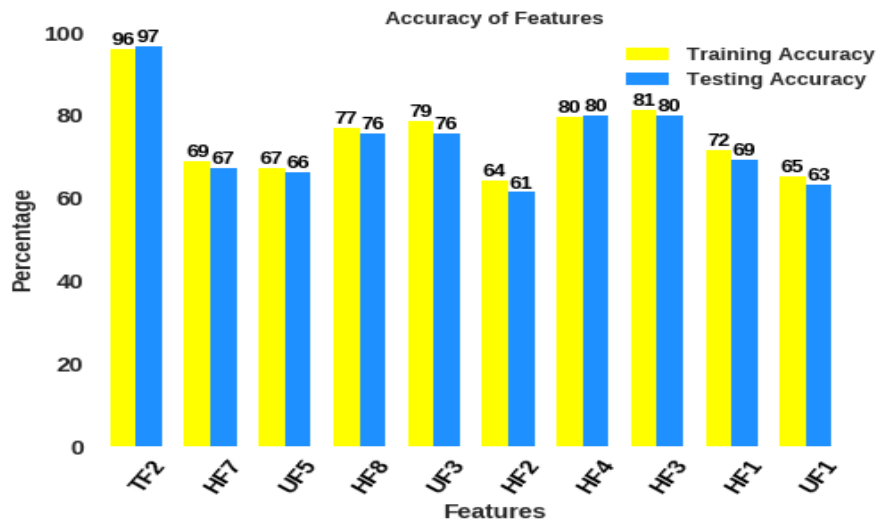


Figure 6.8: LSTM Individual feature accuracy

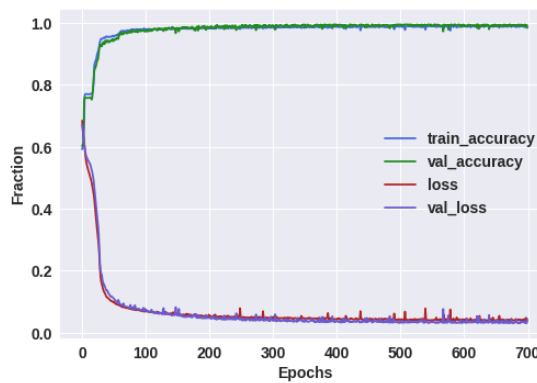


Figure 6.9: Accuracy graph of LSTM

discovered that it did not learn properly due to a lower number of layers (2) by evaluating accuracy and loss value trends during training. So we began by reducing the window size until we reached a window size of 2.

- Stride value: The number of steps skipped after each convolution. Higher strides will reduce the size of the output. We began with a stride value of 4, which reduced the number of convolutional layers in the model. However, the lower convolutional layer performed poorly. As a result, we reduced it to one.
- Number of filters in each layer are tabulated in Table 6.8.

Table 6.8 also includes a list of other hyperparameters. It also performs better with 10

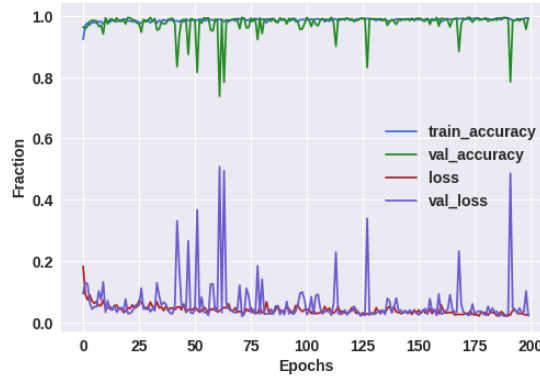


Figure 6.10: Accuracy graph of CNN

features, achieving 99.29% training accuracy and 99.43% testing accuracy. Figure 6.10 depicts the accuracy graph that we obtained.

Table 6.8: Parameters for CNN

Layers	Number of filters in layers	Learning rate	Optimizer	Epochs	Window size	Activation function	Stride
7	32, 64, 64, 128, 128, 264, 512	0.001	Adam Optimizer	200	2	tanh	1

6.4.5 Result analysis

The current investigation encompasses a series of experiments, from feature selection to model validation with the selected heuristics. Three deep learning classifiers are employed to determine the authenticity of websites, distinguishing between phishing and legitimate ones. In the first experiment, individual features are meticulously evaluated using deep neural networks. Out of the initial 18 features, 14 are thoughtfully chosen and subjected to testing with the proposed model. The accuracy of the model is meticulously observed in the second experiment.

The third experiment revolves around streamlining the process by eliminating four underperforming third-party features that significantly prolonged the feature extraction process. Consequently, the best-performing ten features are retained to assess the proposed model. This model comprises three deep learning classifiers and is executed individually, with variations in learning rates, the number of epochs, the architecture of layers, and the number of units in each layer, among other factors. As a result of finely

Table 6.9: Summary of the results of related existing works

Techniques	Accuracy (%)
Zhang et al. (2014)	95.83
Mohammad et al. (2014)	92.48
El-Alfy (2017)	96.79
Zhao et al. (2018)	98.5
Le et al. (2018)	99.29
Bahnsen et al. (2017)	98.7
Yang et al. (2019)	98.99
Feng et al. (2018)	97.71
Yi et al. (2018)	90

tuned experimental parameters, the model exhibits outstanding performance, achieving a remarkable 99.55% training accuracy and 99.52% testing accuracy when employing the DNN model.

Subsequently, the experiments continue with LSTM, employing meticulously fine-tuned parameters, ultimately achieving the highest testing accuracy of 99.57%. Similarly, the exploration extends to CNN, employing the same set of ten features and meticulously fine-tuned parameters, resulting in a testing accuracy of 99.43%. These results conclude that the LSTM model is the most suitable choice for classifying phishing URLs.

6.4.6 Comparison study

In this section, we compare our model with existing works that use deep learning for the classification of phishing sites. Like other researchers (Gowtham and Krishnamurthi 2014; He et al. 2011; Marchal et al. 2017; Ramesh et al. 2014; Yang et al. 2019), the results of existing works are collected from the respective papers for the comparison analysis. The listed results in Table 6.9. are the results obtained by respective authors with their datasets. These researchers' datasets could not be used for comparison because of the limitation of feature extraction. Our technique requires third-party based feature (TF2) for the classification of phishing URLs. The use of third-party services

for feature extraction requires the datasets with live phishing sites. The existing works (listed in Table 6.9) datasets majorly consists of URLs that have already been taken down from the Internet. Hence, they cannot be used for comparison with our work. The comparison study has been conducted with Rao and Pais (2019), CANTINA (Zhang et al. 2007), and CANTINA+ (Xiang et al. 2011) using the common dataset. The results are given in Table 6.10. It is observed that our model with DNN, LSTM, and CNN achieved significant accuracy compared to the existing works. It is also, demonstrated that the proposed model with LSTM outperformed other proposed models with an accuracy of 99.57% which is an improvement over our previous work (99.5%) with minimal features. Note that Le et al. (2018), Bahnsen et al. (2017), and Zhao et al. (2018) applied various deep learning algorithms on the URLs rather than content for the classification of phishing URLs. Le et al. (2018) achieved a significant accuracy compared to other existing works that used features extracted from content (El-Alfy 2017; Feng et al. 2018; Mohammad et al. 2014; Yi et al. 2018; Zhang et al. 2014). Despite the use of content-based features in training our model, it is observed that our model outperformed Le et al. (2018) work and other deep learning based methods of Bahnsen et al. (2017) and Zhao et al. (2018) which use URLs for the classification. This shows the richness of our feature set in detecting the phishing sites.

Table 6.10: Summary of the works implemented on the same dataset.

Techniques	Accuracy (%)
Rao and Pais (2019)	99.5
Zhang et al. (2007)	89.18
Xiang et al. (2011)	99.13
Proposed Model-I [DNN]	99.52
Proposed Model-II [LSTM]	99.57
Proposed Model-III [CNN]	99.43

Deployment of model: The model is deployed as a desktop application that takes URL as an input and gives the status of the URL as output. The application makes a connection to REST API which is running at a remote server where the actual execution of technique takes place. The REST API is hosted on an Intel Xeon 16 core Ubuntu server

with 16GB RAM and a 2.67GHz processor. The REST API is implemented using the Spring framework and the GET method is used to transfer the URL from application to the program running at the remote server. On receiving the URL, the running program at the remote server proceeds with the extraction of features. These features are combined to form a feature vector that is further sent to a trained deep learning model for identifying the legitimacy of the given URL. The REST API sends back the status of the URL (legitimate or phishing) to the application to display the message.

6.5 LIMITATIONS

In this section, we discuss the limitations of our proposed work. Since the proposed model is dependent on third-party services, non availability of these services will limit the performance of our work.

Also, our proposed model might fail to detect phishing sites that use embedded objects such as flash, java scripts and HTML files to replace textual content. In the future, we intend to include the features for the detection of these embedded objects in the phishing sites.

6.6 SUMMARY

In this chapter, we present a deep learning model to detect the legitimacy of a given website. We used URL heuristic and third-party service based features for training the deep learning models. Unlike the work of Rao and Pais (2019), we minimized the number of features and reduced the dependency on third-party services to achieve a significant accuracy of 99.57%. We also tested our features with various deep learning based models such as CNN, DNN & LSTM, and we achieved an accuracy of 99.57% with LSTM, 99.43% with CNN and 99.52% with DNN. The LSTM and DNN outperformed by achieving better results with 10 features than the work Rao and Pais (2019) with machine learning with 18 features.

CHAPTER 7

CONCLUSION AND FUTURE WORK

To conclude, the four research studies introduced novel techniques and models to detect phishing emails and determine website authenticity. The first study suggested using machine learning classifiers and word embedding to detect phishing emails. Using only four email header features, the presented techniques achieved an impressive accuracy rate of 99.50%. The RF classifier performed consistently well with all the word embedding algorithms tested, making it the most appropriate classifier for phishing email classification using word embedding techniques.

The second work presented a novel model, DeepEPhishNet, that combines deep learning and word embedding to classify phishing emails. The model achieved an accuracy of 99.52% for an in-house dataset using only four email header features for classification. These two works may be extended, including incorporating email body features for classification with minimum features, and other researchers may use the in-house dataset for testing their models.

The third work presented a phishing email classification model based on BERT transformers, achieving an accuracy of 99.51% for open-source data. The presented model outperformed all other existing works using only the email body text feature to identify or detect phishing emails. Future work may include extending this work using different advanced transformers and minimum features of the email header and body text for classification.

The fourth research work presented a deep learning model to determine website legitimacy by using URL heuristics and third-party service features to train the models. The presented model attained an impressive accuracy rate of 99.57%, with the LSTM and DNN models performing better than other deep learning models tested. The presented techniques and models contribute significantly to phishing email and website detection.

Future work: Future research in phishing detection can concentrate on several areas to enhance the current studies. Firstly, incorporating additional heuristics using word embedding and machine learning techniques can leverage the content of the email body, leading to improved accuracy in phishing detection. Secondly, evaluating the performance of features through deep learning techniques can further enhance the classification process, resulting in higher accuracy rates. Additionally, utilizing essential features from both email headers and body text can facilitate the development of a more efficient and practical classification model with a minimal feature set. Sharing the in-house dataset with other researchers enables collaborative testing and comparative evaluations, contributing to the advancement of phishing detection techniques. Exploring advanced transformer models that leverage attention mechanisms and contextual representations can also enhance phishing detection capabilities. Furthermore, including additional heuristic features that can enable the detection of phishing sites hosted on compromised domains and identify sophisticated phishing attempts involving embedded objects like iframes, flash, and HTML. By addressing these future directions, the field of phishing detection can make significant strides in the development of more robust and effective techniques to combat online security threats.

BIBLIOGRAPHY

- .
- .
- A Hamid, I. R. and Abawajy, J. (2011). “Hybrid feature selection for phishing email detection.” In *International Conference on Algorithms and Architectures for Parallel Processing*, Springer, 266–275.
- Abu-Nimeh, S., Nappa, D., Wang, X. and Nair, S. (2009). “Distributed phishing detection by applying variable selection using bayesian additive regression trees.” In *2009 IEEE International Conference on Communications*, IEEE, 1–5.
- Adam, S. (2021). “Sophos: Phishing insights 2021.” <https://news.sophos.com/en-us/2021/08/26/phishing-insights-2021/>.
- Adebowale, M. A., Lwin, K. T. and Hossain, M. A. (2023). “Intelligent phishing detection scheme using deep learning algorithms.” *Journal of Enterprise Information Management*, 36(3), 747–766.
- Adewole, K. S., Akintola, A. G., Salihu, S. A., Faruk, N. and Jimoh, R. G. (2019). “Hybrid rule-based model for phishing urls detection.” In *International Conference for Emerging Technologies in Computing*, Springer, 119–135.
- Afek, Y., Bremler-Barr, A. and Shafir, L. (2017). “Network anti-spoofing with sdn data plane.” In *IEEE INFOCOM 2017-IEEE conference on computer communications*, IEEE, 1–9.

BIBLIOGRAPHY

- Al-Hamar, Y., Kolivand, H., Tajdini, M., Saba, T. and Ramachandran, V. (2021). “Enterprise credential spear-phishing attack detection.” *Computers & Electrical Engineering*, 94, 107363.
- Al-Musib, N. S., Al-Serhani, F. M., Humayun, M. and Jhanjhi, N. (2021). “Business email compromise (bec) attacks.” *Materials Today: Proceedings*.
- Alam, M. N., Sarma, D., Lima, F. F., Saha, I., Hossain, S. et al. (2020). “Phishing attacks detection using machine learning approach.” In *2020 third international conference on smart systems and inventive technology (ICSSIT)*, IEEE, 1173–1179.
- Aldowah, H., Ul Rehman, S. and Umar, I. (2018). “Security in internet of things: issues, challenges and solutions.” In *International conference of reliable information and communication technology*, Springer, 396–405.
- Alhogail, A. and Alsabih, A. (2021). “Applying machine learning and natural language processing to detect phishing email.” *Computers & Security*, 110, 102414.
- Almeida, F. and Xexéo, G. (2019). “Word embeddings: A survey.” *arXiv preprint arXiv:1901.09069*.
- Almomani, A., Gupta, B. B., Atawneh, S., Meulenberg, A. and Almomani, E. (2013). “A survey of phishing email filtering techniques.” *IEEE communications surveys & tutorials*, 15(4), 2070–2090.
- Appel, G., Grewal, L., Hadi, R. and Stephen, A. T. (2020). “The future of social media in marketing.” *Journal of the Academy of Marketing Science*, 48(1), 79–95.
- APWG (2014). “Apwg 2014 global phishing reports first half 2014.” https://docs.apwg.org/reports/APWG_Global_Phishing_Report_1H_2014.pdf. Accessed: 2014-09-25.
- APWG (2018). “Apwg 2018 phishing attack trends reports, first quarter 2018.” https://docs.apwg.org/reports/apwg_trends_report_q1_2018.pdf. Accessed: 2018-07-31.

- APWG (2019a). “Apwg 2018 phishing attack trends reports, fourth quarter 2018..” https://docs.apwg.org/reports/apwg_tre-nds_report_q4_2018.pdf. Accessed: 2019-03-04.
- APWG (2019b). “Apwg 2019 phishing activity trends reports, third quarter 2019..” https://docs.apwg.org//reports/apwg_tre-nds_report_q3_2019.pdf. Accessed: 2019-11-04.
- Aravindhana, R., Shanmugalakshmi, R. and Ramya, K. (2017). “Circumvention of nascent and potential wi-fi phishing threat using association rule mining.” *Wireless Personal Communications*, 94(4), 2331–2361.
- Athulya, A. and Praveen, K. (2020). “Towards the detection of phishing attacks.” In *2020 4th international conference on trends in electronics and informatics (ICOEI)(48184)*, IEEE, 337–343.
- Azeez, N. A., Misra, S., Margaret, I. A., Fernandez-Sanz, L. et al. (2021). “Adopting automated whitelist approach for detecting phishing attacks.” *Computers & Security*, 108, 102328.
- Bagui, S., Nandi, D., Bagui, S. and White, R. J. (2019). “Classifying phishing email using machine learning and deep learning.” In *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, IEEE, 1–2.
- Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J. and González, F. A. (2017). “Classifying phishing urls using recurrent neural networks.” In *2017 APWG symposium on electronic crime research (eCrime)*, IEEE, 1–8.
- Balamurugan, G. and Jayabharathy, J. (2022). “Cyberbully classification based on tweet texts for detection of phishing links.” In *Smart Data Intelligence*, Springer, 367–374.
- Balim, C. and Gunal, E. S. (2019). “Automatic detection of smishing attacks by machine learning methods.” In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, IEEE, 1–3.

- Baykara, M. and Gürel, Z. Z. (2018). “Detection of phishing attacks.” In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, IEEE, 1–5.
- 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.
- Biron, Z. A., Dey, S. and Pisu, P. (2018). “Real-time detection and estimation of denial of service attack in connected vehicle systems.” *IEEE Transactions on Intelligent Transportation Systems*, 19(12), 3893–3902.
- Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017). “Enriching word vectors with subword information.” *Transactions of the Association for Computational Linguistics*, 5, 135–146.
- Bountakas, P., Koutroumpouchos, K. and Xenakis, C. (2021). “A comparison of natural language processing and machine learning methods for phishing email detection.” In *The 16th International Conference on Availability, Reliability and Security*, 1–12.
- Buber, E., Diri, B. and Sahingoz, O. K. (2017). “Nlp based phishing attack detection from urls.” In *International Conference on Intelligent Systems Design and Applications*, Springer, 608–618.
- Burns, A., Johnson, M. E. and Caputo, D. D. (2019). “Spear phishing in a barrel: Insights from a targeted phishing campaign.” *Journal of Organizational Computing and Electronic Commerce*, 29(1), 24–39.
- Castillo, E., Dhaduvai, S., Liu, P., Thakur, K.-S., Dalton, A. and Strzalkowski, T. (2020). “Email threat detection using distinct neural network approaches.” In *Proceedings for the First International Workshop on Social Threats in Online Conversations: Understanding and Management*, 48–55.
- Chahid, Y., Benabdellah, M. and Azizi, A. (2017). “Internet of things security.” In *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, IEEE, 1–6.

- Chandrasekaran, M., Narayanan, K. and Upadhyaya, S. (2006). “Phishing email detection based on structural properties.” In *NYS cyber security conference*, volume 3, Albany, New York.
- Chavan, S., Inamdar, A., Dorle, A., Kulkarni, S. and Wu, X.-W. (2020). “Phishing detection: malicious and benign websites classification using machine learning techniques.” In *Proceeding of International Conference on Computational Science and Applications*, Springer, 437–446.
- Chiew, K. L., Tan, C. L., Wong, K., Yong, K. S. and Tiong, W. K. (2019). “A new hybrid ensemble feature selection framework for machine learning-based phishing detection system.” *Information Sciences*, 484, 153–166.
- Choi, H. S., Carpenter, D. and Ko, M. S. (2022). “Risk taking behaviors using public wi-fi™.” *Information Systems Frontiers*, 24(3), 965–982.
- Cohen, A., Nissim, N. and Elovici, Y. (2018). “Novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods.” *Expert Systems with Applications*, 110, 143–169.
- Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding.” *arXiv preprint arXiv:1810.04805*.
- Do, V. T., Engelstad, P., Feng, B. and Van Do, T. (2017). “Detection of dns tunneling in mobile networks using machine learning.” In *International Conference on Information Science and Applications*, Springer, 221–230.
- Dudheria, R. (2017). “Evaluating features and effectiveness of secure qr code scanners.” In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, IEEE, 40–49.
- Eder-Neuhauser, P., Zseby, T., Fabini, J. and Vormayr, G. (2017). “Cyber attack models for smart grid environments.” *Sustainable Energy, Grids and Networks*, 12, 10–29.

- El-Alfy, E.-S. M. (2017). “Detection of phishing websites based on probabilistic neural networks and k-medoids clustering.” *The Computer Journal*, 60(12), 1745–1759.
- Fang, X., Xu, M., Xu, S. and Zhao, P. (2019a). “A deep learning framework for predicting cyber attacks rates.” *EURASIP Journal on Information security*, 2019(1), 1–11.
- Fang, Y., Zhang, C., Huang, C., Liu, L. and Yang, Y. (2019b). “Phishing email detection using improved rnn model with multilevel vectors and attention mechanism.” *IEEE Access*, 7, 56329–56340.
- Feng, F., Zhou, Q., Shen, Z., Yang, X., Han, L. and Wang, J. (2018). “The application of a novel neural network in the detection of phishing websites.” *Journal of Ambient Intelligence and Humanized Computing*, 1–15.
- Fette, I., Sadeh, N. and Tomasic, A. (2007). “Learning to detect phishing emails.” In *Proceedings of the 16th international conference on World Wide Web*, 649–656.
- Fu, A. Y., Wenyin, L. and Deng, X. (2006). “Detecting phishing web pages with visual similarity assessment based on earth mover’s distance (emd).” *IEEE transactions on dependable and secure computing*, 3(4), 301–311.
- Gansterer, W. N. and Pölz, D. (2009). “E-mail classification for phishing defense.” In *European Conference on Information Retrieval*, Springer, 449–460.
- Garcés, I. O., Cazares, M. F. and Andrade, R. O. (2019). “Detection of phishing attacks with machine learning techniques in cognitive security architecture.” In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 366–370.
- Ghimire, A., Jha, A. K., Thapa, S., Mishra, S. and Jha, A. M. (2021). “Machine learning approach based on hybrid features for detection of phishing urls.” In *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, IEEE, 954–959.
- Gowtham, R. and Krishnamurthi, I. (2014). “A comprehensive and efficacious architecture for detecting phishing webpages.” *Computers & Security*, 40, 23–37.

- Guarda, T., Augusto, M. F. and Lopes, I. (2019). “The art of phishing.” In *International conference on information technology & systems*, Springer, 683–690.
- Guillén, J. H., Del Rey, A. M. and Casado-Vara, R. (2019). “Security countermeasures of a sciras model for advanced malware propagation.” *IEEE Access*, 7, 135472–135478.
- Gupta, S. and Singhal, A. (2017). “Phishing url detection by using artificial neural network with pso.” In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)*, IEEE, 1–6.
- Gutierrez, C. N., Kim, T., Della Corte, R., Avery, J., Goldwasser, D., Cinque, M. and Bagchi, S. (2018). “Learning from the ones that got away: Detecting new forms of phishing attacks.” *IEEE Transactions on Dependable and Secure Computing*, 15(6), 988–1001.
- 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*, Springer, 266–275.
- Hara, M., Yamada, A. and Miyake, Y. (2009). “Visual similarity-based phishing detection without victim site information.” In *2009 IEEE Symposium on Computational Intelligence in Cyber Security*, IEEE, 30–36.
- Harikrishnan, N., Vinayakumar, R. and Soman, K. (2018). “A machine learning approach towards phishing email detection.” In *Proceedings of the Anti-Phishing Pilot at ACM International Workshop on Security and Privacy Analytics (IWSPA AP)*, volume 2013, 455–468.
- Hasan, M., Balbahaith, Z. and Tarique, M. (2019). “Detection of sql injection attacks: a machine learning approach.” In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, IEEE, 1–6.
- He, M., Horng, S.-J., Fan, P., Khan, M. K., Run, R.-S., Lai, J.-L., Chen, R.-J. and Sutanto, A. (2011). “An efficient phishing webpage detector.” *Expert systems with applications*, 38(10), 12018–12027.

- Heartfield, R. and Loukas, G. (2018). “Protection against semantic social engineering attacks.” In *Versatile Cybersecurity*, Springer, 99–140.
- Higbee, A. (2021). “Cofense: Annual state of phishing report-2021..”
)<https://cofense.com/wp-content/uploads/2021/02/cofense-annual-report-2021.pdf>.
- Hiransha, M., Unnithan, N. A., Vinayakumar, R., Soman, K. and Verma, A. (2018). “Deep learning based phishing e-mail detection.” In *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*, Tempe, AZ, USA.
- Hochreiter, S. and Schmidhuber, J. (1997). “Long short-term memory.” *Neural computation*, 9(8), 1735–1780.
- Hu, Y., Chen, L. and Cheng, J. (2018). “A captcha recognition technology based on deep learning.” In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, IEEE, 617–620.
- Huh, J. H. and Kim, H. (2011). “Phishing detection with popular search engines: Simple and effective.” In *International Symposium on Foundations and Practice of Security*, Springer, 194–207.
- Islam, R. and Abawajy, J. (2013). “A multi-tier phishing detection and filtering approach.” *Journal of Network and Computer Applications*, 36(1), 324–335.
- J Kuss, D., D Griffiths, M., Karila, L. and Billieux, J. (2014). “Internet addiction: A systematic review of epidemiological research for the last decade.” *Current pharmaceutical design*, 20(25), 4026–4052.
- Jain, A. K. and Gupta, B. (2018a). “Phish-safe: Url features-based phishing detection system using machine learning.” In *Cyber Security*, Springer, 467–474.
- Jain, A. K. and Gupta, B. B. (2018b). “Towards detection of phishing websites on client-side using machine learning based approach.” *Telecommunication Systems*, 68(4), 687–700.

- Jain, A. K. and Gupta, B. B. (2018c). “Two-level authentication approach to protect from phishing attacks in real time.” *Journal of Ambient Intelligence and Humanized Computing*, 9(6), 1783–1796.
- Jamal, T., Haider, Z., Butt, S. A. and Chohan, A. (2018). “Denial of service attack in cooperative networks.” *arXiv preprint arXiv:1810.11070*.
- Jia, J., Dong, Z., Li, J. and Stokes, J. W. (2021). “Detection of malicious dns and web servers using graph-based approaches.” In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2625–2629.
- Jonker, R. A. A., Poudel, R., Pedrosa, T. and Lopes, R. P. (2021). “Using natural language processing for phishing detection.” In *International Conference on Optimization, Learning Algorithms and Applications*, Springer, 540–552.
- Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H. and Mikolov, T. (2016a). “Fasttext. zip: Compressing text classification models.” *arXiv preprint arXiv:1612.03651*.
- Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2016b). “Bag of tricks for efficient text classification.” *arXiv preprint arXiv:1607.01759*.
- Jozefowicz, R., Zaremba, W. and Sutskever, I. (2015). “An empirical exploration of recurrent network architectures.” In *International conference on machine learning*, PMLR, 2342–2350.
- Kaltiala-Heino, R., Lintonen, T. and Rimpelä, A. (2004). “Internet addiction? potentially problematic use of the internet in a population of 12–18 year-old adolescents.” *Addiction Research & Theory*, 12(1), 89–96.
- Kaspersky (2019). “Spam and phishing in q3 2019.” <https://securelist.com/spam-report-q3-2019/95177/>).

- Kathrine, G. J. W., Praise, P. M., Rose, A. A. and Kalaivani, E. C. (2019). “Variants of phishing attacks and their detection techniques.” In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, 255–259.
- 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), 40.
- Khonji, M., Iraqi, Y. and Jones, A. (2013). “Phishing detection: a literature survey.” *IEEE Communications Surveys & Tutorials*, 15(4), 2091–2121.
- Khonji, M., Jones, A. and Iraqi, Y. (2011). “A study of feature subset evaluators and feature subset searching methods for phishing classification.” In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, 135–144.
- Kim, J.-Y., Bu, S.-J. and Cho, S.-B. (2018). “Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders.” *Information Sciences*, 460, 83–102.
- Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2017). “Imagenet classification with deep convolutional neural networks.” *Communications of the ACM*, 60(6), 84–90.
- Kumar, M. S. and Indrani, B. (2021). “Frequent rule reduction for phishing url classification using fuzzy deep neural network model.” *Iran Journal of Computer Science*, 4(2), 85–93.
- Kunju, M. V., Dainel, E., Anthony, H. C. and Bhelwa, S. (2019). “Evaluation of phishing techniques based on machine learning.” In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, IEEE, 963–968.
- Le, H., Pham, Q., Sahoo, D. and Hoi, S. C. (2018). “Urlnet: Learning a url representation with deep learning for malicious url detection.” *arXiv preprint arXiv:1802.03162*.

- Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G. and Wolff, S. S. (1997). “The past and future history of the internet.” *Communications of the ACM*, 40(2), 102–108.
- Li, Q., Cheng, M., Wang, J. and Sun, B. (2020). “Lstm based phishing detection for big email data.” *IEEE transactions on big data*, 8(1), 278–288.
- Li, Y., Yang, Z., Chen, X., Yuan, H. and Liu, W. (2019). “A stacking model using url and html features for phishing webpage detection.” *Future Generation Computer Systems*, 94, 27–39.
- Ma, L., Ofoghi, B., Watters, P. and Brown, S. (2009). “Detecting phishing emails using hybrid features.” In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, IEEE, 493–497.
- Malaysia, U. (2013). “An enhanced online phishing e-mail detection framework based on evolving connectionist system.” *Int. J. Innov. Comput., Inf. Control*, 9(3), 1065–1086.
- Marchal, S., Armano, G., Gröndahl, T., Saari, K., Singh, N. and Asokan, N. (2017). “Off-the-hook: An efficient and usable client-side phishing prevention application.” *IEEE Transactions on Computers*, 66(10), 1717–1733.
- Maurya, S. and Jain, A. (2020). “Deep learning to combat phishing.” *Journal of Statistics and Management Systems*, 23(6), 945–957.
- Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013). “Efficient estimation of word representations in vector space.” *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., Joulin, A., Chopra, S., Mathieu, M. and Ranzato, M. (2014). “Learning longer memory in recurrent neural networks.” *arXiv preprint arXiv:1412.7753*.
- Miloslavskaya, N. and Tolstoy, A. (2017). “Ensuring information security for internet of things.” In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, IEEE, 62–69.

BIBLIOGRAPHY

- Mimecast (2019). “Mimecast - the state of the email security report 2019..” <https://www.mimecast.com/globalassets/documents/ebook/state-of-email-security-2019.pdf?v=2>.
- 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.
- Mohammadi, M., Chu, B. and Lipford, H. R. (2017). “Detecting cross-site scripting vulnerabilities through automated unit testing.” In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 364–373.
- Mondal, S., Ghosh, S., Kumar, A., Islam, S. H. and Chatterjee, R. (2022). “Spear phishing detection: An ensemble learning approach.” In *Data Analytics, Computational Statistics, and Operations Research for Engineers*, CRC Press, 203–234.
- Moradpoor, N., Clavie, B. and Buchanan, B. (2017). “Employing machine learning techniques for detection and classification of phishing emails.” In *2017 Computing Conference*, IEEE, 149–156.
- Nagunwa, T., Naqvi, S., Fouad, S. and Shah, H. (2019). “A framework of new hybrid features for intelligent detection of zero hour phishing websites.” In *International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on European Transnational Education (ICEUTE 2019)*, Springer, 36–46.
- Nathezhtha, T., Sangeetha, D. and Vaidehi, V. (2019). “Wc-pad: web crawling based phishing attack detection.” In *2019 International Carnahan Conference on Security Technology (ICCST)*, IEEE, 1–6.
- Nguyen, M., Nguyen, T. and Nguyen, T. H. (2018). “A deep learning model with hierarchical lstms and supervised attention for anti-phishing.” *arXiv preprint arXiv:1805.01554*.
- Oña, D., Zapata, L., Fuertes, W., Rodríguez, G., Benavides, E. and Toulkeridis, T. (2019). “Phishing attacks: detecting and preventing infected e-mails using machine

- learning methods.” In *2019 3rd Cyber Security in Networking Conference (CSNet)*, IEEE, 161–163.
- Opara, C., Wei, B. and Chen, Y. (2020). “Htmlphish: enabling phishing web page detection by applying deep learning techniques on html analysis.” In *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 1–8.
- Parsons, K., Butavicius, M., Delfabbro, P. and Lillie, M. (2019). “Predicting susceptibility to social influence in phishing emails.” *International Journal of Human-Computer Studies*, 128, 17–26.
- Patil, V., Thakkar, P., Shah, C., Bhat, T. and Godse, S. (2018). “Detection and prevention of phishing websites using machine learning approach.” In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, IEEE, 1–5.
- Peng, T., Harris, I. and Sawa, Y. (2018). “Detecting phishing attacks using natural language processing and machine learning.” In *2018 IEEE 12th international conference on semantic computing (icsc)*, IEEE, 300–301.
- Pham, C., Nguyen, L. A., Tran, N. H., Huh, E.-N. and Hong, C. S. (2018). “Phishing-aware: A neuro-fuzzy approach for anti-phishing on fog networks.” *IEEE Transactions on Network and Service Management*, 15(3), 1076–1089.
- Pham, N.-Q., Kruszewski, G. and Boleda, G. (2016). “Convolutional neural network language models.” In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1153–1162.
- Quinlan, J. R. (1986). “Induction of decision trees.” *Machine learning*, 1(1), 81–106.
- Ra, V., HBa, B. G., Ma, A. K., KPa, S., Poornachandran, P. and Verma, A. (2018). “Deepanti-phishnet: Applying deep neural networks for phishing email detection.” In *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*, Tempe, AZ, USA, 1–11.

- Ramanathan, V. and Wechsler, H. (2012). “phishigillnet-phishing detection methodology using probabilistic latent semantic analysis, adaboost, and co-training.” *EURASIP Journal on Information Security*, 2012(1), 1–22.
- Ramesh, G., Krishnamurthi, I. and Kumar, K. S. S. (2014). “An efficacious method for detecting phishing webpages through target domain identification.” *Decision Support Systems*, 61, 12–22.
- Rao, R. S. and Ali, S. T. (2015). “A computer vision technique to detect phishing attacks.” In *2015 Fifth International Conference on Communication Systems and Network Technologies*, IEEE, 596–601.
- Rao, R. S. and Pais, A. R. (2019). “Detection of phishing websites using an efficient feature-based machine learning framework.” *Neural Computing and Applications*, 31(8), 3851–3873.
- Rashid, J., Mahmood, T., Nisar, M. W. and Nazir, T. (2020). “Phishing detection using machine learning technique.” In *2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, IEEE, 43–46.
- Ryan, T., Chester, A., Reece, J. and Xenos, S. (2014). “The uses and abuses of facebook: A review of facebook addiction.” *Journal of behavioral addictions*, 3(3), 133–148.
- Rybakov, O. J. and Rybakova, O. S. (2019). “Principles of information security of a child on the internet.” In *Ubiquitous Computing and the Internet of Things: Prerequisites for the Development of ICT*, Springer, 427–433.
- Saha, I., Sarma, D., Chakma, R. J., Alam, M. N., Sultana, A. and Hossain, S. (2020). “Phishing attacks detection using deep learning approach.” In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, IEEE, 1180–1185.
- Sahingoz, O. K., Buber, E., Demir, O. and Diri, B. (2019). “Machine learning based phishing detection from urls.” *Expert Systems with Applications*, 117, 345–357.

- Salem, O., Alsubhi, K., Shaafi, A., Gheryani, M., Mehaoua, A. and Boutaba, R. (2021). “Man-in-the-middle attack mitigation in internet of medical things.” *IEEE Transactions on Industrial Informatics*, 18(3), 2053–2062.
- Salem, O., Hossain, A. and Kamala, M. (2010). “Awareness program and ai based tool to reduce risk of phishing attacks.” In *2010 10th IEEE International Conference on Computer and Information Technology*, IEEE, 1418–1423.
- Salihovic, I., Serdarevic, H. and Kevric, J. (2018). “The role of feature selection in machine learning for detection of spam and phishing attacks.” In *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*, Springer, 476–483.
- Schuster, M. and Paliwal, K. K. (1997). “Bidirectional recurrent neural networks.” *IEEE transactions on Signal Processing*, 45(11), 2673–2681.
- Shaw, M. and Black, D. W. (2008). “Internet addiction.” *CNS drugs*, 22(5), 353–365.
- Singh, S., Singh, M. and Pandey, R. (2020). “Phishing detection from urls using deep learning approach.” In *2020 5th international conference on computing, communication and security (ICCCS)*, IEEE, 1–4.
- Smadi, S., Aslam, N. and Zhang, L. (2018). “Detection of online phishing email using dynamic evolving neural network based on reinforcement learning.” *Decision Support Systems*, 107, 88–102.
- Smith, C. and Jin, Y. (2014). “Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction.” *Neurocomputing*, 143, 302–311.
- Somesha, M. and Pais, A. R. (2022). “Classification of phishing email using word embedding and machine learning techniques.” *Journal of Cyber Security and Mobility*, 279–320.
- Somesha, M., Pais, A. R., Rao, R. S. and Rathour, V. S. (2020). “Efficient deep learning techniques for the detection of phishing websites.” *Sādhanā*, 45(1), 1–18.

- Sonowal, G. (2022a). “Phishing kits.” In *Phishing and Communication Channels*, Springer, 115–135.
- Sonowal, G. (2022b). “Types of phishing.” In *Phishing and Communication Channels*, Springer, 25–50.
- Sontowski, S., Gupta, M., Chukkapalli, S. S. L., Abdelsalam, M., Mittal, S., Joshi, A. and Sandhu, R. (2020). “Cyber attacks on smart farming infrastructure.” In *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, IEEE, 135–143.
- Srinivasa Rao, R. and Pais, A. R. (2017). “Detecting phishing websites using automation of human behavior.” In *Proceedings of the 3rd ACM workshop on cyber-physical system security*, 33–42.
- Storm, B. C., Stone, S. M. and Benjamin, A. S. (2017). “Using the internet to access information inflates future use of the internet to access other information.” *Memory*, 25(6), 717–723.
- Taib, R., Yu, K., Berkovsky, S., Wiggins, M. and Bayl-Smith, P. (2019). “Social engineering and organisational dependencies in phishing attacks.” In *IFIP Conference on Human-Computer Interaction*, Springer, 564–584.
- Toolan, F. and Carthy, J. (2009). “Phishing detection using classifier ensembles.” In *2009 eCrime researchers summit*, IEEE, 1–9.
- Toolan, F. and Carthy, J. (2010). “Feature selection for spam and phishing detection.” In *2010 eCrime Researchers Summit*, IEEE, 1–12.
- Ulfath, R. E., Sarker, I. H., Chowdhury, M. J. M. and Hammoudeh, M. (2022). “Detecting smishing attacks using feature extraction and classification techniques.” In *Proceedings of the International Conference on Big Data, IoT, and Machine Learning*, Springer, 677–689.
- Valecha, R., Mandaokar, P. and Rao, H. R. (2021). “Phishing email detection using persuasion cues.” *IEEE Transactions on Dependable and Secure Computing*.

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017). “Attention is all you need.” *Advances in neural information processing systems*, 30.
- Verma, P., Goyal, A. and Gigras, Y. (2020). “Email phishing: Text classification using natural language processing.” *Computer Science and Information Technologies*, 1(1), 1–12.
- Verma, R., Shashidhar, N. and Hossain, N. (2012). “Detecting phishing emails the natural language way.” In *European Symposium on Research in Computer Security*, Springer, 824–841.
- Wenyin, L., Huang, G., Xiaoyue, L., Min, Z. and Deng, X. (2005). “Detection of phishing webpages based on visual similarity.” In *Special interest tracks and posters of the 14th international conference on World Wide Web*, 1060–1061.
- Whittaker, C., Ryner, B. and Nazif, M. (2010). “Large-scale automatic classification of phishing pages.” .
- Wollschlaeger, M., Sauter, T. and Jasperneite, J. (2017). “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0.” *IEEE industrial electronics magazine*, 11(1), 17–27.
- Xiang, G., Hong, J., Rose, C. P. and Cranor, L. (2011). “Cantina+ a feature-rich machine learning framework for detecting phishing web sites.” *ACM Transactions on Information and System Security (TISSEC)*, 14(2), 1–28.
- Yadollahi, M. M., Shoeleh, F., Serkani, E., Madani, A. and Gharaee, H. (2019). “An adaptive machine learning based approach for phishing detection using hybrid features.” In *2019 5th International Conference on Web Research (ICWR)*, IEEE, 281–286.
- Yang, P., Zhao, G. and Zeng, P. (2019). “Phishing website detection based on multidimensional features driven by deep learning.” *IEEE access*, 7, 15196–15209.

- Yao, W., Ding, Y. and Li, X. (2018). “Deep learning for phishing detection.” In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, IEEE, 645–650.
- Yerima, S. Y. and Alzaylaee, M. K. (2020). “High accuracy phishing detection based on convolutional neural networks.” In *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, IEEE, 1–6.
- Yi, P., Guan, Y., Zou, F., Yao, Y., Wang, W. and Zhu, T. (2018). “Web phishing detection using a deep learning framework.” *Wireless Communications and Mobile Computing*, 2018.
- Yuan, H., Chen, X., Li, Y., Yang, Z. and Liu, W. (2018). “Detecting phishing websites and targets based on urls and webpage links.” In *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 3669–3674.
- Zabihimayvan, M. and Doran, D. (2019). “Fuzzy rough set feature selection to enhance phishing attack detection.” In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 1–6.
- Zhang, D., Yan, Z., Jiang, H. and Kim, T. (2014). “A domain-feature enhanced classification model for the detection of chinese phishing e-business websites.” *Information & Management*, 51(7), 845–853.
- Zhang, N. and Yuan, Y. (2012). “Phishing detection using neural network.” *CS229 lecture notes*, 301.
- Zhang, W., Jiang, Q., Chen, L. and Li, C. (2017). “Two-stage elm for phishing web pages detection using hybrid features.” *World Wide Web*, 20(4), 797–813.
- Zhang, Y., Hong, J. I. and Cranor, L. F. (2007). “Cantina: a content-based approach to detecting phishing web sites.” In *Proceedings of the 16th international conference on World Wide Web*, 639–648.

- Zhang, Y.-y., Chen, J.-j., Ye, H. and Volantin, L. (2020). “Psychological effects of cognitive behavioral therapy on internet addiction in adolescents: A systematic review protocol.” *Medicine*, 99(4).
- Zhao, J., Wang, N., Ma, Q. and Cheng, Z. (2018). “Classifying malicious urls using gated recurrent neural networks.” In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Springer, 385–394.
- Zinovyeva, E., Härdle, W. K. and Lessmann, S. (2020). “Antisocial online behavior detection using deep learning.” *Decision Support Systems*, 138, 113362.

PUBLICATIONS

1. Somesha, M., Pais, A. R., Rao, R. S., & Rathour, V. S. (2020). **Efficient deep learning techniques for the detection of phishing websites..** *Sadhana (Springer)*, 45(1), 1-18. [**Published**]
2. Somesha, M., & Pais, A. R. (2022). **Classification of Phishing Email Using Word Embedding and Machine Learning Techniques.** *Journal of Cyber Security and Mobility, (River publisher)*, 279-320. [**Published**]
3. Somesha, M., & Pais, A. R. (2022). **DeepEPhishNet: A Deep learning framework for email phishing detection using Word embedding algorithms.** *Journal of Cyber Security and Mobility, (River publisher)*. [**Accepted**]
4. Somesha, M., & Pais, A. R. (2022). **Phishing Classification based on Text Content of an Email Body using Transformers.** *International Conference on Information Security, Privacy and Digital Forensics (Springer)* [**Accepted**]

BIO-DATA

Name: Somesha M

Date of Birth: 01/05/1974

Gender: Male

Marital Status: Married

Father's Name: Hanuma Naik M

Mother's Name: Late. Shankamma M

Wife Name: Shashikala R

Email Id: somesh.naik@gmail.com

Present Address: Assistant Professor,
Dept. of Computer Science & Engineering,
Govt. SKSJ Technological Institute, K R Circle,
Bangalore – 560001.

Educational Qualification: B.E (CSE) – UVCE, Bangalore – 560001
M,Tech (Computer Network Engineering),
National Institute of Engineering, Mysore – 570006.

Areas of Interest: Information Security, Cyber Security, Computer Networks,
Phishing, Machine Learning.