ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

Phishing Attack Detection Based on Deep Learning and Natural Language Processing

THESIS SUBMITTED AS PART OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY IN INFORMATICS

DIEGO EDUARDO BENAVIDES ASTUDILLO

diego.benavides@epn.edu.ec

SUPERVISOR: WALTER MARCELO FUERTES DÍAZ, Ph.D.

walter.fuertes@epn.edu.ec

CO-SUPERVISOR: SANDRA SÁNCHEZ GORDÓN, Ph.D.

sandra.sanchez@epn.edu.ec

Quito, july 2024

ESCUELA POLITÉCNICA NACIONAL

THESIS

For the award of the degree of

DOCTOR OF PHILOSOPHY IN INFORMATICS

Resolution RPC-SO-43-No.501-2014

of the Consejo de Educación Superior

Presented by

DIEGO EDUARDO BENAVIDES ASTUDILLO

Thesis supervised by

WALTER MARCELO FUERTES DÍAZ Ph.D.,

Professor of the National Polytechnic School (Ecuador)

and co-supervised by

SANDRA SÁNCHEZ GORDÓN Ph.D.,

Professor of the National Polytechnic School (Ecuador)

Phishing Attack Detection Based on Deep Learning and Natural Language Processing

Oral examination

by the following committee:

Myriam Hernández, Ph.D.

Escuela Politécnica Nacional (EPN - Ecuador)

Denys Flores, Ph.D.

Escuela Politécnica Nacional (EPN - Ecuador)

Luis Urquiza, Ph.D.

Escuela Politécnica Nacional (EPN - Ecuador)

Enrique Vinicio Carrera, Ph.D.

Universidad de las Fuerzas Armadas (ESPE - Ecuador)

José Javier Martínez, Ph.D.

Universidad de Alcalá (España)

DECLARATION

I, Diego Eduardo Benavides Astudillo, declare under oath that I am the author of this work, which has not previously been presented for obtaining an academic degree or professional qualification. I also declare that I have consulted the bibliographic references included in this document.

Through this declaration, I transfer my intellectual property rights corresponding to this thesis to the Escuela Politécnica Nacional, as established by the Intellectual Property Law of Ecuador, its Regulations, and the current institutional norms.

I declare that this work is based on the following articles of my authorship as primary author related to the title of this thesis:

Journals:

- Eduardo Benavides-Astudillo et al., "NDLP Phishing: A Fine-tuned Application to Detect Phishing Attacks Based on Natural Language Processing and Deep Learning," in International Journal of Interactive Mobile Technologies, vol. 18, pp. 173-190, 2024 [1].
- Eduardo Benavides-Astudillo et al., "A Phishing-Attack-Detection Model Using Natural Language Processing and Deep Learning," in Applied Sciences, vol. 13, p. 5275, 2023 [2].

Conferences:

- Eduardo Benavides-Astudillo et al., "Classification of Phishing Attack Solutions by Employing Deep Learning Techniques: A Systematic Literature Review," in Smart Innovation, Systems and Technologies, vol. 152, 2020 [3].

- Eduardo Benavides-Astudillo et al., "Comparative Study of Deep Learning Algorithms in the Detection of Phishing Attacks Based on HTML and Text Obtained from Web Pages," in Communications in Computer and Information Science, vol. 1755, pp. 386–398, 2022 [4].
- Eduardo Benavides-Astudillo et al., "A Framework Based on Personality Traits to Identify Vulnerabilities to Social Engineering Attacks," in Communications in Computer and Information Science, vol. 394, no. 1535, p. 381, 2022 [5].
- Eduardo Benavides-Astudillo et al., "Analysis of Vulnerabilities Associated with Social Engineering Attacks Based on User Behavior," in Communications in Computer and Information Science, vol. 1535, pp. 351-364, 2022 [6].

I also declare that I have acknowledged the collaboration of third parties and the contribution made by other published or unpublished material.

DIEGO EDUARDO BENAVIDES ASTUDILLO

CERTIFICATION

I certify that Diego Eduardo Benavides Astudillo has conducted his research under my supervision. To the best of my knowledge, the contributions of this work are novel.

Firmado electrónicamente por:
WALTER MARCELO
FUERTES DIAZ

Walter Marcelo Fuertes Díaz, Ph.D.

SUPERVISOR

Firmado electrónicamente por:
SANDRA PATRICIA
SANCHEZ GORDON

Sandra Patricia Sánchez Gordón, Ph.D.

CO SUPERVISOR

DEDICATORY

I want to dedicate this work to my mother, Elena Astudillo, who has been the cornerstone of my academic journey with her unwavering love and sacrifice. Your constant support and wise counsel have been my anchor in times of uncertainty and my beacon in the darkest days.

To my father, Diego Benavides (+), who, despite adversity, always tried to guide me along the path of good. Who, as a fundamental principle of life, instilled in me that it doesn't matter what I am as long as I am a good person.

To my beloved wife, Jenny Zabala, whose love, strength, and optimism have been a beacon of hope on my path to completing this project. Your constant encouragement and faith in me have reminded me that I can reach for the stars if I dare to chase them.

To my children, Diego, Sara, and Axel Benavides Zabala, who are my reason for living and who have endured my physical and mental absence, but to whom, with this example, I want to show that there is no unattainable goal and that perseverance and sacrifice are the main attitudes for achieve any goal.

To my esteemed mentors, Walter Fuertes and Sandra Sánchez, your profound wisdom and guidance have enriched my intellect and shaped my character. Your unwavering commitment to academic excellence and passion for knowledge have been a perpetual source of inspiration.

Finally, to all those who have contributed to this work in one way or another, whether with their collaboration, comments or simply their presence, I sincerely thank you. This achievement would not have been possible without your generosity and support.

With humility and gratitude, I dedicate this work to all of you.

ACKNOWLEDGMENTS

First and foremost, I am indebted to my thesis director, Walter Fuertes, and my co-director, Sandra Sánchez. Their expert guidance, constant encouragement, and unwavering support have been pivotal to the success of this doctoral thesis. Their unique perspectives, vast experience, and dedicated mentorship have shaped this work and inspired me to push my boundaries.

I extend my heartfelt gratitude to the members of my doctoral committee, Myriam Hernández, Denys Flores, Luis Urquiza, Vinicio Carrera, and José Javier Martínez. Their valuable time, insightful comments, and expert guidance during the preparation of this thesis have been instrumental in its development. Their diverse perspectives and rigorous review have significantly enhanced the quality of this work.

I want to acknowledge the generous support of the Universidad de las Fuerzas Armadas ESPE. Your support has been essential to carrying out this research. Additionally, I want to thank the Escuela Politécnica Nacional for granting me this opportunity for personal growth. Also, my sincere gratitude goes to my research colleagues in the Department of Computer Sciences at ESPE Santo Domingo. They have shared their knowledge and experiences and created a collaborative and enriching environment to develop this work.

I cannot overlook the invaluable support of my friends and family, who have been by my side every step, providing encouragement, understanding, and unconditional love. Your support has been my rock in times of challenge and doubt.

In short, I am deeply grateful to everyone who has contributed to this project in any way. Their support and encouragement have been instrumental in this journey, and I will always remember them with gratitude.

PROLOGUE

My motivation for carrying out this thesis was to offer a model and an application so that people can have an efficient tool that uses the latest technologies to detect phishing attacks. This model uses Deep Learning and natural language processing technologies to detect malicious text on phishing web pages.

While numerous solutions employ Deep Learning to detect Phishing attacks, they often overlook the wealth of information embedded in web page text. Conversely, most Deep Learning-based detection techniques focus solely on web page URLs, leaving a significant gap in the field.

Due to this knowledge gap, my objective was to create a model that uses Deep Learning and Natural Language Processing with the GloVe dictionary to detect phishing attacks on web page text, taking advantage of that text's semantic and syntactic richness.

To achieve this objective, I embarked on a comprehensive journey. We began with a thorough literature review, followed by an evaluation of phishing attack victims' personality traits and behaviors. Then, I delved into the significance of HTML code and text. Next, I developed and tested the proposed model with four deep-learning algorithms. Finally, I fine-tuned and deployed an application of the proposed model.

This work required a lot of effort, which would have been impossible without the support of my advisors, Dr. Walter Fuertes and Dr. Sandra Sánchez. I also thank the Escuela Politécnica Nacional (EPN) and the Universidad de las Fuerzas Armadas (ESPE) for their logistical support and the resources for this research.

It has been an arduous and long journey to reach this pinnacle; however, I feel great satisfaction at having completed this challenge, which has planted in me the seed to research and continue investigating, intending to give back to life and society a part of my joys received.

Table of Contents

List of Figures

List of Tables

RESUMEN

Phishing es un tipo de ciberataque de Ingeniería Social que tiene como objetivo engañar a los usuarios finales, normalmente utilizando páginas web. El método más común para detectar a este tipo de ataques es por medio de comparar las direcciones URLs con una blacklist de URLs ya identificadas como páginas de phishing. Sin embargo, el principal problema es cuando aparecen páginas de phishing nuevas no registradas en la blacklist. Actualmente, una de las formas más comunes de detectar estas páginas de phishing no identificadas con anterioridad, es analizando el contenido de las páginas web, es decir, ingresando palabras de forma no secuencial en algoritmos de aprendizaje profundo, sin importar la secuencia del texto ingresado en los algoritmos de Deep Learning. El objetivo general de esta tesis es proponer un modelo que detecte ataques de phishing basándose en el texto de páginas web sospechosas, utilizando Deep Learning, Procesamiento de Lenguaje Natural y Word Embedding con GloVe dictionary. De esta forma aprovechamos la riqueza semántica y sintáctica del texto de la página analizada. Para lograr nuestro objetivo se realizó una revisión de la literatura, se evaluaron los rasgos de personalidad y comportamiento de las personas, se implementó y afinó el modelo de detección de phishing, y finalmente se hizo una extensión en Chrome llamada NDLP para detectar estos ataques. Se determinó que el modelo funciona pues los cuatro algoritmos evaluados LSTM, BiLSTM, GRU, and BiGRU obtuvieron sobre el 96.70% de mean accuracy, y que el algoritmo que dio mejores resultados fue BiGRU que logró 97.39%. de mean accuracy.

Palabras Claves - Phishing, Ingeniería Social, Aprendizaje Profundo, Procesamiento de Lenguaje Natural, Diccionario GloVe, Incrustación de Palabras.

ABSTRACT

Phishing is a type of Social Engineering cyber attack that aims to deceive end users, usually using web pages. The most common method to detect this type of attack is by comparing the URLs with a blacklist of URLs already identified as phishing pages. However, the main problem is when new phishing pages appear that are not registered on the blacklist. Currently, one of the most common ways to detect these previously unidentified phishing pages is by analyzing the content of the web pages, that is, by entering words non-sequentially into deep learning algorithms, regardless of the sequence of the text entered in Deep Learning algorithms. The main objective of this thesis is to propose a model that detects phishing attacks based on the text of suspicious web pages, using Deep Learning, Natural Language Processing, and Word Embedding with the GloVe dictionary. In this way, we take advantage of the semantic and syntactic richness of the text on the analyzed page. To achieve the main objective, we conducted a literature review, evaluated people's personality and behavioral traits, and implemented, evaluated, and refined the phishing detection model. Finally, we made a Chrome extension called NDLP to detect these attacks. It was determined that the model works because the four evaluated algorithms, LSTM, BiLSTM, GRU, and BiGRU, obtained over 96.70% mean accuracy, and the algorithm that gave the best results was BiGRU, which achieved 97.39%.

Keywords - Phishing, Social Engineering, Deep Learning, Natural Language Processing, GloVe Dictionary, Word Embedding.

Chapter 1

Introduction

Contents

1.1. Problem Statement

Social Engineering is a cyber attack in which attackers try to trick end users into obtaining confidential information, primarily to defraud them. One of the main types of Social Engineering attacks is through a phishing web page. Attackers create a phishing web page that appears legitimate. This phishing web page may appear from a bank, credit card, or online store.

Currently, the primary and most straightforward method to confront this type of attack is through a blacklist of phishing web pages. The blacklist of phishing web pages has stored the URLs of the previously identified pages. This method is very effective if a page has already been identified as phishing. The problem with this method is that new pages are created every day, and therefore they are not registered on this blacklist.

Machine Learning algorithms have been implemented to identify these new pages. These algorithms can predict whether a page is highly likely to be phishing based on similar characteristics to those detected in previous pages. There are different traditional Machine Learning solutions to address this type of attack, but Deep Learning algorithms stand out among them. Deep Learning is a branch of Machine Learning, but it differs from traditional algorithms because it has deeper layers. Deep Learning translates into better accuracy in detecting phishing attacks.

In the state-of-the-art review, most solutions proposed to mitigate phishing that uses Deep Learning to analyze the URLs of web pages, leaving aside the text in the web pages. On the other hand, the solutions found do not carry out a semantic and syntactic analysis of the text obtained from the web pages. In other words, they do not use Natural Language Processing.

1.2. Justification

In a computing environment, there are many methods to protect users from cyberattacks. These protection methods can be hardware, software, or user consent when Internet browsing. However, a chain will always be broken by its weakest link, which in many cases is the end user, because it is enough for an attacker to request the personal data of a victim because they have a job offer or because the victim has won the lottery so that this user finally ends up falling victim to this attack.

It should be noted that Phishing attacks are mostly non-targeted; that is, they are not directed at a single person but at many people trying to get one of the people who read a phishing email or web page to take the bait. Precisely, the term Phishing refers to the fishing of people.

We justify this work because we understand that to many people it is straightforward to detect a Social Engineering attack, specifically a Phishing attack, such as observing if a page being opened has a lock in the URL bar or even reading if the URL that was opened is what you want to access. On the other hand, of this large number of people, some groups are more vulnerable and fall victim to these attacks. For this reason, we carried out this research to offer a tool that uses the latest technology and is available to vulnerable people to detect if the page that opens in the browser is Phishing.

People are generally victims mainly of their personality traits or behavior. When we talk about personality traits, we mean that some people are more trusting than others or have

a helpful spirit, among other features. In contrast, when discussing behavior, we refer to people who browse the entire web as careless and risky. According to [7], in terms of the relationship between personality and behavior, the Big Five Model suggests that individual differences in these personality dimensions influence how people think, feel, and behave in various situations.

To analyze and determine the most vulnerable personality traits, we wrote the article *A Framework Based on Personality Traits to Identify Vulnerabilities to Social Engineering Attacks* [5], whereas to determine the risky behavior of people, we wrote the article *Analysis of vulnerabilities associated with Social Engineering attacks based on user behavior* [6]. The content of the two articles can be seen in Annex A.

1.3. Objectives

1.3.1. General Objective

The main objective of this work is to offer users a high-precision tool that uses the latest technologies for detecting phishing attacks based on Deep Learning and Natural Language Processing.

It was necessary to carry out the following secondary objectives to achieve the general objective.

1.3.2. Specific Objectives

- Conduct a Systematic Review of the Literature on Implementing Deep Learning to detect and mitigate Phishing attacks.
- Conduct a comparative study of the Deep Learning algorithms used to detect Phishing attacks.
- Determine a Phishing attack detection model using Deep Learning and NLP.
- Tune Deep Learning and NLP hyper-parameters in a detection model and implement an application to detect Phishing attacks.

1.4. Scientific Contribution

This research offers a refined model from which an application was made to detect phishing attacks using Deep Learning and NLP. Thus, the application takes advantage of the syntactic and semantic richness of the text extracted from phishing web pages. Using this application is enough for the end user to install an extension in their browser, which will inform them of the percentage probability a page has of being phished. We made the following scientific contributions to achieve our objective.

To create a new model to detect phishing attacks, it was necessary to establish the characteristics of these attacks and how current models mitigate them. Thus, in the article *Classification of Phishing Attack Solutions by Employing Deep Learning Techniques: A Systematic Literature Review* [3], the general and particular characteristics of the attacks are first established. Then, a complete review was conducted on how each Deep Learning algorithm is applied and which datasets and other resources were used to mitigate Phishing attacks.

In the study [3], it was determined that most solutions that implement Deep Learning are oriented to the URLs of web pages, which is why there is still much to study in approaches oriented to the content of web pages. This is why it is justified to carry out a study that analyzes with which content of web pages the best results are obtained, with HTML code or with only the text extracted from that code. Additionally, previous observations revealed that many Deep Learning algorithms currently detect phishing attacks. However, it is necessary to determine which of these algorithms is most accurate in detecting attacks by analyzing their content. To answer all these questions, the article *Comparative Study of Deep Learning Algorithms in the Detection of Phishing Attacks Based on HTML and text Obtained from Web Pages* [4] was carried out. This study determined the following findings:

- There is no significant difference between whether the Deep Learning algorithm runs on HTML code or just the text extracted from that HTML code.
- It was determined that BiGRU is the Deep Learning algorithm that performs best among the LSTM, BiLSTM, GRU, and BiGRU algorithms for text analysis.

As stated by Kevin Mitnick [8], there are undoubtedly users who are more prone than others to fall victim to social engineering attacks. This vulnerability is increased by two main things: the users' personalities or their careless behavior. As part of our study, we set out

to evaluate users' most common personality and behavioral characteristics to fall victim to social engineering attacks. For this, we conducted the following two investigations:

A Framework Based on Personality Traits to Identify Vulnerabilities to Social Engineering Attacks [5] and *Analysis of Vulnerabilities Associated with Social Engineering Attacks Based on User Behavior* [6]. The article [5] evaluated the characteristics that users meet, compared to the Five-Factor Model (FFM) frame. The study determined that the most common personality traits for a user to fall victim to a phishing attack are Openness (28.8%), followed by Agreeableness (27.9%), Conscientiousness (20.2%), Neuroticism (11.5%), and Extraversion (11.5%) respectively. The article [6] evaluated the behavior of people that makes them more likely to be victims of social engineering attacks.

1.5. Other Scientific Contributions

Since this study began, the author has contributed as an author or co-author to other works to combat phishing attacks. These related works of the author can be seen in Table 1.1 and Table 1.2.

Table 1.1: Other related scientific contributions - Part 1.

Article	Journal	Authors	Year	DOI
Un experimento para crear conciencia en las personas acerca de los ataques de Ingeniería Social [9]	Ciencia UNEMI	Benavides-Astudillo Eduardo, Fuertes-Díaz Walter, Sánchez-Gordon Sandra	2020	10.29076/issn.2528-7737
Caracterización de los ataques de phishing y técnicas para mitigarlos: una revisión sistemática de la literatura [10]	Ciencia y Tecnología	Benavides-Astudillo Eduardo, Fuertes-Díaz Walter, Sánchez-Gordon Sandra, Nuñez-Agurto Daniel	2020	10.18779/cyt.v13i1.357
Phishing Attack Detection: A Solution Based on the Typical Machine Learning Modeling Cycle [11]	International Conference on Computational Science and Computational Intelligence (CSCI)	Espinoza Bryan, Simba Jéssica, Fuertes Walter, Benavides Eduardo, Andrade Roberto, Toulkeridis Theofilos	2019	10.29076/issn.2528-7737

Table 1.2: Other related scientific contributions - Part 2.

Article	Journal	Authors	Year	DOI
Phishing Attacks: Detecting and Preventing Infected E-mails Using Machine Learning Methods [12]	Cyber Security in Networking Conference (CSNet)	Oña Diego, Zapata Lenín, Fuertes Walter, Rodríguez Germán, Benavides Eduardo, Toulkeridis Theofilos	2019	10.1109/CSNet47905.2019.9108961
Trusted Phishing: A Model to Teach Computer Security Through the Theft of Cookies [13]	Advances in Emerging Trends and Technologies	Rodríguez Germán, Torres Jenny, Flores Pamela, Benavides Eduardo, Proaño Paola	2019	10.1007/978-3-030-32033-1_36
Impact of Social Engineering Attacks: A Literature Review [14]	Developments and Advances in Defense and Security	Fuertes Walter, Arévalo Diana, Castro Joyce Denisse, Ron Mario, Estrada Carlos Andrés, Andrade Roberto, Peña Felix Fernández, Benavides Eduardo	2021	10.1007/978-981-16-4884-7_3

1.6. Organization of the Document

The rest of this document is structured as follows: Chapter 2 shows the key concepts applied in this research. Chapter 3 reviews the literature about the methods that use Deep Learning to mitigate phishing attacks. In Chapter 4, four Deep Learning algorithms are evaluated with both HTML code and clear text obtained from that HTML code to determine which types of input data to the algorithms, HTML or text, are most effective in detecting phishing attacks. Chapter 5 presents the steps to carry out the Deep Learning and natural language processing model, which is the central axis of this work. In Chapter 6, the proposed model is implemented and tested, and it is determined which is the best algorithm among the four Deep Learning algorithms analyzed. In Chapter 7, the model obtained in the previous chapter is refined, and an extension is made to install it in the Chrome browser. Finally, Chapter 8 shows the conclusions and future work of this research.

Chapter 2

Background

Contents

2.1. Social Engineering

According to Kevin Mitnick, in his book *The art of the deception* [8], Social Engineering refers to manipulating people to obtain confidential information or perform specific actions that can compromise security. To achieve their goal, attackers could include impersonating someone they trust through phone calls, emails, or even in person to persuade people to reveal sensitive information.

2.1.1. Phishing

As first described in 1987 [15], Phishing is a technique of imitating an authoritative entity to deceive users into stealing confidential and susceptible information. Later, in 1996, a Usenet

newsgroup redefined the term Phishing by placing itself around stealing accounts and passwords in American Online (AOL) (https://www.phishing.org/history-of-phishing) [15]. Phishing is a form of cyber attack in which attackers use deceptive tactics, such as fake emails, text messages, or websites, to impersonate legitimate entities to obtain victims' sensitive information, such as usernames, passwords, or financial information [16].

Phishing Life Cycle

In general, the life cycle of a Phishing attack consists of five consecutive steps (see Fig. 2.1) [17]:

1. The phisher creates a fraudulent web page. This website is usually a copy of a popular benign website.
2. The phisher tries to lure users to the malicious page using various means, such as email, WhatsApp, Messenger, and other web pages.
3. Once users are on the fraudulent page, they are asked for confidential information. Later, this information is used against the same users or their organizations.
4. The attack is consummated. In other words, money or other benefits are obtained from the users without them realizing it is illicit.
5. The phisher deletes all evidence that allows it to be traced.

Figure 2.1: Phishing attack life-cycle

Types of Phishing Attacks

Phishing attacks can be grouped into seven categories [18]:

Deceptive Phishing. Through this attack, the phisher sends many deceptive e-mails with links that redirect the user to spoofed web pages.

Malware-based Phishing. Attachments are sent to the victim differently. These files contain malicious code that runs on the user's computer as soon as they are downloaded.

Key Loggers and Screen Loggers. It is similar to the previous attack. In this attack, the phisher installs and uses keyloggers and screen loggers to capture typed information and screenshots and then sends them to the attacker.

Session Hijacking. Through this attack, valid user sessions are manipulated to gain unauthorized access to information stored on a user's computer.

Web Trojans. They appear unnoticed when users try to enter their credentials on a computer. With the help of these web trojans, the user's credentials are collected locally and transmitted to the phisher.

DNS-based Phishing. Through this attack, users are redirected to a fraudulent site, altering the user's host file or the domain name of a company they intend to visit.

Man-in-the-middle Phishing. First, hackers position themselves between the user and the legitimate website. Later, they record the information entered but continue transmitting it so that the users' transactions are unaffected. Finally, they send that information to themselves.

It is worth noting that most Phishing attacks are carried out on websites or e-mails [19], trying to imitate as accurately as possible legitimate websites and e-mails. Web page-based approaches are divided into three groups [20]: (i) approaches based on the URL address of the web page [21], (ii) approaches based on the content of the web page [22], and (iii) approaches based on the visual appearance of the images [23]. As can be observed in Figure 2.2, detecting a Phishing page is relatively easy for an expert in the field; it will notice that instead of using "oo" in the Facebook word in the URL, the phisher will place "00"; however, this detection is challenging for common users [24].

On the other hand, email-based Phishing approaches can be classified into two types [25]: (i) approaches oriented to the e-mail header and (ii) approaches based on the e-mail body as illustrated in Figure 2.3.

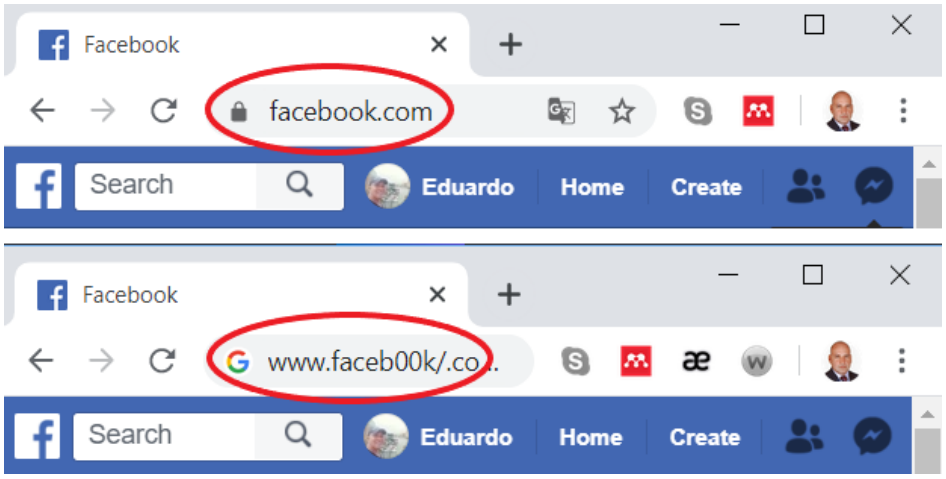


Figure 2.2: Example of website Phishing on URL

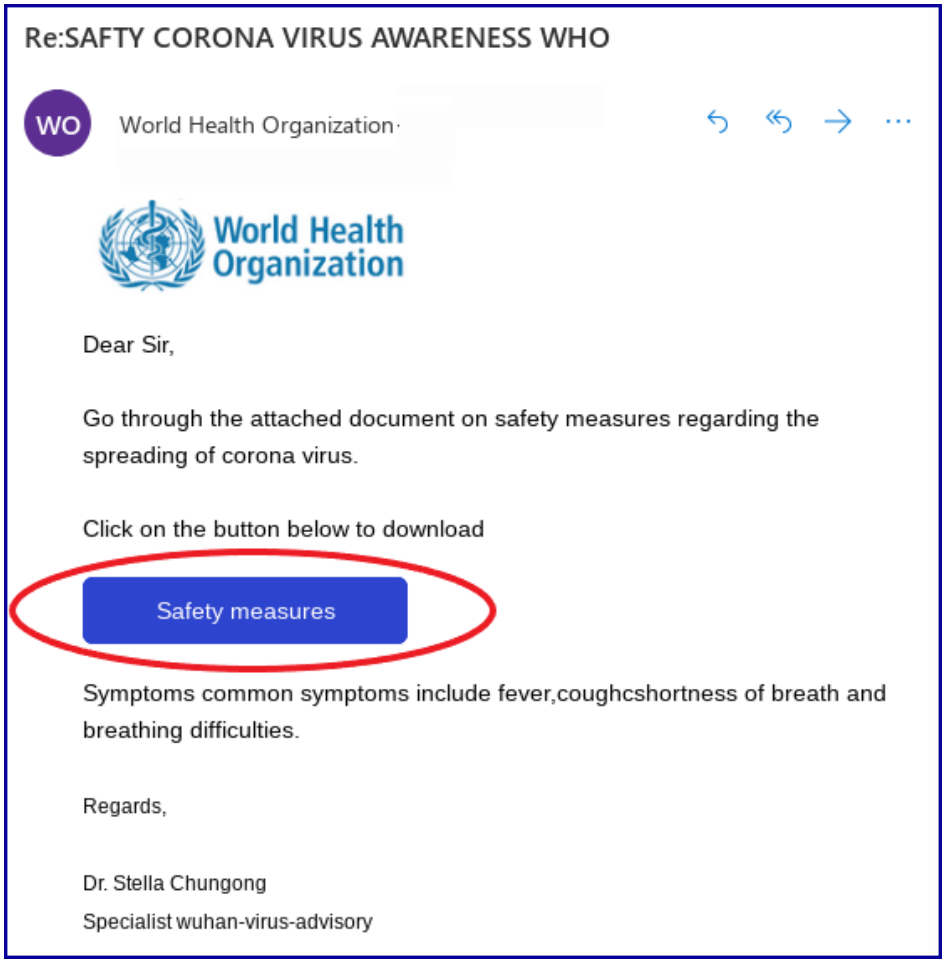


Figure 2.3: Example of e-mail Phishing

Anti-phishing Techniques

The methods used for detecting Phishing pages can be divided into three [26], [27]:

- Blacklist-based Phishing detection methods;
- Heuristic Phishing detection methods; and
- ML-based Phishing detection methods.

From the methods exposed, Machine Learning is the most effective method for the early detection of Phishing attacks. However, it is worth distinguishing between traditional Machine Learning algorithms and algorithms based on Deep Learning. The methods for finding the relevant characteristics of a data set in traditional Machine Learning may involve considerable time consumption for the people involved in finding these features [28]. One of the critical aspects of adopting a Deep Learning-based approach is that it is unnecessary to select characteristics by hand. Instead, Deep Learning can select essential characteristics by discovering an intricate structure in analyzing a large amount of data [29]. In [30], an experiment demonstrates the higher accuracy of Deep Learning algorithms over traditional ML algorithms. Additionally, it is observed that the training time in Deep Learning is much longer (80 times difference); however, the memory usage is much lower (almost 500 times difference).

2.1.2. Web Page-Based Phishing Mitigation Approaches

According to [19], web page-based approaches are divided into three groups:

- Approaches based on the URL address of the web page [20].
- Approaches based on the web page's content [21].
- Approaches based on the visual appearance of the images [22].

2.1.3. Email-Based Phishing Mitigation Approaches

According to [23], the email-based solution approaches can be divided into two groups: Approaches oriented to the email header, and those based on the email body.

2.2. Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence that aims to develop models that allow computers to learn independently without being specifically programmed. The

learning is carried out based on the data entered. This learning is based on data and previous experiences [31].

The idea that machines learn comes from around 1950, from the hand of Alan Touring [32]. Although he did not establish the concept of Machine Learning, he did lay the foundations for developing Artificial Intelligence and Machine Learning. Turing addressed the fundamental question: Can machines think? The idea of Touring is that machines can learn based on their experience, by trial and error, and thus improve their performance without the need to follow a predefined algorithm.

2.2.1. Deep Learning

According to [33], Deep Learning is a Machine Learning technique where many layers of information processing stations are exploited by classification patterns and features or by learning by representation. However, it has become popular due to two factors: a notable increase in processing capabilities (e.g., video cards, graphical processors, etc.) and the large amount of data available.

An advantage of Deep Learning over traditional Machine Learning is that the user or expert doesn't need to indicate the features of a malicious page because the algorithm can detect them automatically and independently [34]. Another advantage is that the more data that is processed in a Deep Learning algorithm, the greater accuracy is obtained, while in a traditional Machine Learning algorithm, a plateau is reached, from which a better result can no longer be obtained [35].

Depending on whether the algorithms are trained to yield obtainable results, Machine Learning and Deep Learning algorithms' Phishing solutions may be classified into three subgroups: unsupervised, supervised, and hybrid. Our study is of a supervised type and binary classification because it is based on detecting whether the text of a web page is malicious based on pre-trained algorithms with a class that indicates whether it is Phishing or ham.

Deep Learning is a subset of ML techniques where there are two or more internal and hidden layers, hierarchically organized, to process information. See Figure 2.4. The main advantages of using Deep Learning techniques instead of traditional ML techniques are twofold: (i) Deep learning techniques give more accurate results than traditional ML techniques, and (ii) Traditional ML techniques are very time-consuming, dedicated to manually selecting the best features for the detection of a Phishing threat, whereas in Deep Learn-

ing this process is automatic, that is, the Deep Learning algorithm selects by itself the best features.

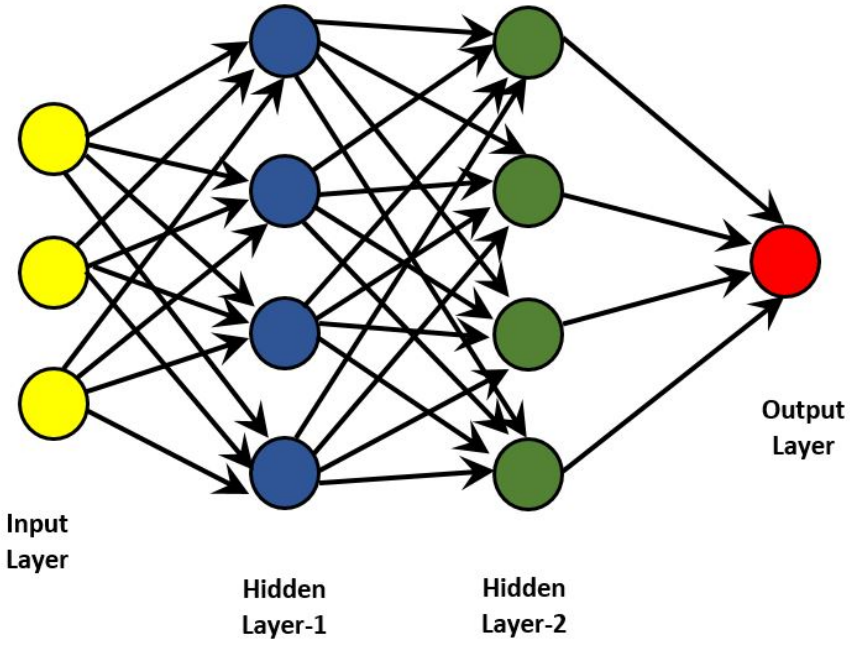


Figure 2.4: Example of a Deep Neural Network with two hidden layers

Fig. 2.4 depicts the general scheme of a Deep Neural Network (DNN) [33]. Thus, in the input layer, the characteristics vector of the object analyzed is entered, while in the output layer, there is a probability vector associated with the given inputs. Between these two vectors, there are many layers (two layers, with blue and green nodes in this graph) whose nodes or neurons are interconnected. Thus, the output of the computation of one neuron is the input of the next neuron.

2.3. Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence that combines computational linguistics and Artificial Intelligence to enable computers to understand and interpret natural human language [36].

It is stated in the article [2] that the majority of attack mitigation techniques that use Deep Learning do not perform a semantic or syntactic analysis of the text of the analyzed web pages; the text captured from the pages is entered sequentially into the different Deep Learning algorithms, thus disregarding the intrinsic richness in the relationships between the

entered words. For this reason, the model proposed in this study is oriented towards first pre-processing the text using Natural Language Processing.

2.4. Global Vectors for Word Representation

Global Vectors for Word Representation (GloVe) is a vector representation model commonly used with Natural Language Processing. The main idea is that when entering the text to be analyzed, the semantic and syntactic relationships between the words of the text are captured, analyzing the co-occurrences of words in large volumes of data. Thus, each word is assigned a vector, and in this way, by subtracting or performing operations on these vectors, the existing relationships between the words can be obtained [37].

2.5. Chapter Summary

This chapter reviewed the fundamental concepts necessary to understand this research work. The approach used goes from general to particular, starting with Social Engineering concepts and clarifying that Phishing is the most frequent type of Social Engineering cyber attack. It is highlighted that the best way to detect a new Phishing attack is by using Deep Learning. However, the precision when applying a Deep Learning algorithm in text detection improves when fed with a dataset previously processed with Natural Language Processing. However, the algorithm can improve even more if we embed the input text with the GloVe Dictionary, through which the semantic relationships and similarities between the words are captured. Furthermore, at this point, it is also defined that the problem to be solved is binary and that the algorithms used for the analyses in the following chapters are supervised.

Chapter 3

Literature Review

Contents

For this thesis, two literature reviews were carried out. The first literature review was carried out at the beginning of this research. It was aimed solely at knowing the most common techniques for detecting attacks with Phishing and deep learning, how these techniques were applied, and above all, knowing the gaps in the investigation. One of the most important findings of this first review highlights the little treatment given to implementing NLP and deep learning for detecting Phishing attacks. The second literature review was carried out in the middle of our research and was aimed at studying phishing mitigation methods using deep learning techniques, which NLP also supported. This second literature review helped place our proposal among other related works.

3.1. Literature Review of Phishing and Deep Learning

This first Systematic Literature Review (SLR) was conducted to determine the Deep Learning techniques used to combat phishing attacks and the data and tools used in each study. The methodology proposed by Barbara Kitchenham [38] was rigorously followed to carry out this Systematic Literature Review. First, the search was conducted on five digital scientific bases (ACM, IEEExplore, Scopus, SpringerLink, and Taylor & Francis), and 29 primary studies were selected. From each of the primary studies selected, the following attributes were characterized: year of publication, scientific database, the country to which it belongs, type of paper, number of references, and accuracy, among others. Subsequently, the synthesis of the techniques used was fulfilled, and the studies were classified by attack used, Deep Learning techniques and sub-techniques used, repositories used, the accuracy obtained, and the amount of data used.

3.1.1. Research Methodology

We followed the Systematic Literature Review methodology proposed by Barbara Kitchenham [38], in which five consecutive steps were conducted:

1. Define the research questions;
2. Search the relevant documents;
3. Select the primary studies;
4. Map selected primary studies.

First, the related articles were searched for; then, the relevant results were listed. Then, data extraction was performed, and the reports of the information encountered were finally produced. Each phase is described next.

Define Research Questions

Main Question

Initially, we define our research's main research question (MQ). This MQ is: What Deep Learning techniques are currently used in primary studies to mitigate Phishing attacks?

Secondary Questions Additionally, we define the following secondary research questions (RQ).

- RQ1. What are the causes and consequences of Phishing attacks on society?
- RQ2. What Deep Learning techniques are currently used in primary studies to mitigate Phishing attacks?
- RQ3. How do researchers use Deep Learning techniques or algorithms to reduce the impact of Phishing attacks?

The first question, RQ1, provides an overview of the causes and consequences of cyber-attacks on society. The second question, RQ2, identifies which Deep Learning techniques have been used to detect and mitigate these attacks. Finally, the third question, RQ3, lets us know how each method has been implemented in primary studies to reduce its impact.

Search Relevant Documents

First, our strategy consisted of defining the search string including the general terms Phishing and Deep Learning. Then, the search string was refined to include specific Deep Learning techniques. Thus, the search string is as follows:

- Phishing and ("Deep Learning" or Autoencoder or "Sum-Product Network" or "Recurrent Neural Network" or "Boltzmann Machine" or "Deep Neural Network" or "Convolutional Neural Network").

Inclusion Criteria With the research questions defined, we only included the following publications:

- Whose main topics are Phishing and Deep Learning and its techniques;
- Research with results;
- Writing only in English;
- Describe the application of at least one Deep Learning technique for detecting Phishing attacks;
- Studies published from 2017 onwards.

Exclusion Criteria We excluded the following publications:

- Articles about Phishing only or Deep Learning only, but not both.

- Articles written in any language other than English.
- Research without precision results.
- Research that does not apply at least one of the Deep Learning techniques.

Quality Criteria To comply with the quality criteria of the search, it has been decided to conduct the scrutiny only on sources recognized at the scientific level, which are also from information technology. The indexed databases chosen are: IEEE Explore, Scopus, Taylor and Francis, Springer, ACM Library, and Science Direct.

Select Primary Studies

Once the search string was applied to digital databases, 180 publications were initially found. Nevertheless, after eliminating duplicated articles, reading abstracts, reading all the content, and applying the inclusion and exclusion criteria, only 45 primary studies were selected, as illustrated in Fig. 3.1.

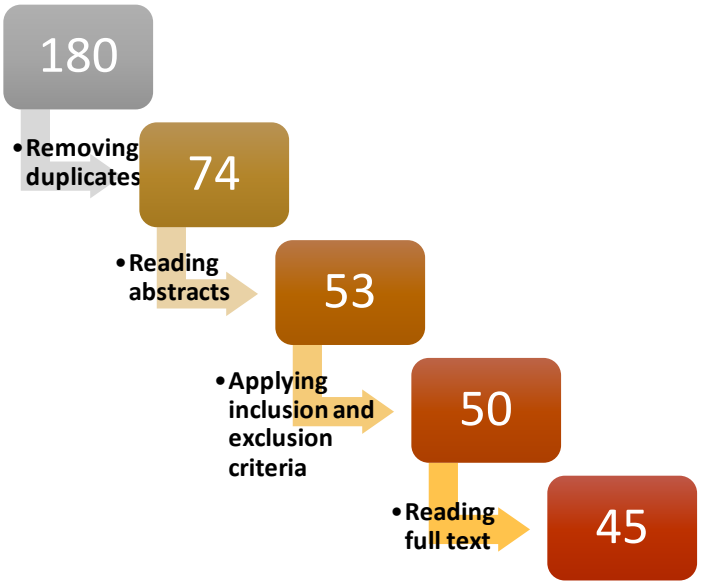


Figure 3.1: Search and selection process

Table 3.1 details information regarding each primary study, such as the title, the database in which they were found, what type of publication are (Conference publication, Journal, Book series), the quartile assigned in the Scimago Journal & Country Rank (SJR), (NQ=No quartile) according to SJR, the total number of references, the publication year, and the country to which the work belongs.

Table 3.1: Selected primary studies.

No.	Title	Ref.	Database	publication type	SJR Quartile	Total Refs.	Year	Country
1	Phishing Email Detection Using Improved RCNN Model with Multilevel Vectors and Attention Mechanism	[25]	Scopus	Journal	Q1	51	2019	China
2	Phishing Website Detection Based on Multidimensional Features Driven by deep learning	[39]	Web of Science	Journal	Q1	38	2019	China
3	Evaluating deep learning approaches to characterize and classify malicious URL's	[40]	Web of Science	Journal	Q2	27	2018	India
4	Personalized, Browser-Based Visual Phishing Detection Based on deep learning	[41]	IEEE Xplore	Book Series	Q2	6	2019	Italy
5	Phishing Detection Method Based on Borderline-Smote Deep Belief Network	[42]	IEEE Xplore	Book Series	Q2	11	2017	China
6	PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks	[43]	Web of Science	Journal	Q2	34	2019	China
7	Ensemble one-vs-all learning technique with emphatic & rehearsal training for phishing email classification using psychology	[44]	Web of Science	Journal	Q2	60	2018	United States
8	Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting	[45]	Scopus	Journal	Q2	50	2019	Saudi Arabia
9	A deep learning model with hierarchical LSTMs and supervised attention for anti-phishing	[46]	Springer link	Conference publication	Q3	39	2018	Vietnam, United States

10	A platform for automatic identification of phishing URLs in mobile text messages	[47]	Springer Link	Journal	Q3	15	2018	China
11	A.R.E.S: Automatic rogue email spotter crypt coyotes	[48]	Springer Link	Journal	Q3	35	2018	India
12	CNN-based malicious user detection in social networks	[49]	Springer Link	Journal	Q3	23	2018	South Korea
13	Deep learning-based phishing E-mail detection CEN-Deepspam	[19]	Scopus	Journal	Q3	20	2018	India
14	Web phishing detection using a deep learning framework	[50]	Web of Science	Journal	Q3	79	2018	China, USA
15	Phishing Analysis of Websites Using Classification Techniques	[51]	IEEE Xplore	Book Series	Q3	18	2019	Turkey
16	Phishing Detection Research Based on LSTM Recurrent Neural Network	[52]	Scopus	Book Series	Q3	19	2018	China
17	Deep belief network-based detection and categorization of malicious URLs	[34]	Scopus	Journal	Q3	52	2018	India
18	A Deep-Learning-Driven Light-Weight Phishing Detection Sensor	[24]	Web of Science	Journal	Q3	26	2019	United Kingdom
19	Comparison of Ensemble Simple Feedforward Neural Network and Deep Learning Neural Network on Phishing Detection	[53]	Scopus	Conference publication	Q3	23	2020	Malaysia
20	Parameter Setting for Deep Neural Networks Using Swarm Intelligence on Phishing Websites Classification	[54]	Scopus	Journal	Q3	50	2019	Slovenia
21	Advanced Phishing Filter Using Autoencoder and Denoising Autoencoder	[55]	ACM Library	Conference publication	NQ	27	2017	Morocco

22	Hunting Malicious TLS Certificates with Deep Neural Networks	[56]	ACM Library	Conference publication	NQ	40	2018	Colombia
23	Swarm Intelligence Approaches for Parameter Setting of Deep Learning Neural Network: Case Study on Phishing Websites Classification	[57]	ACM Library	Conference publication	NQ	34	2018	Slovenia
24	LSTM based self-defending AI chatbot providing anti-phishing	[58]	ACM Library	Conference publication	NQ	47	2018	India
25	Chrome Extension for Malicious URLs detection in Social Media Applications Using Artificial Neural Networks and Long Short Term Memory Networks	[59]	IEEE Xplore	Conference publication	NQ	16	2018	India
26	Classification of URL bitstreams using bag of bytes	[60]	IEEE Xplore	Conference publication	NQ	6	2018	Japan
27	Classifying phishing URLs using recurrent neural networks	[30]	Scopus	Conference publication	NQ	6	2017	Colombia
28	Comparative Study of the Detection of Malicious URLs Using Shallow and Deep Networks	[61]	Scopus	Conference publication	NQ	18	2018	India
29	Deep Learning for Phishing Detection	[62]	Scopus	Conference publication	NQ	22	2018	China
30	Defending Internet of Things Against Malicious Domain Names using D-FENS	[63]	Scopus	Conference publication	NQ	44	2018	United States
31	Detecting Homoglyph Attacks with a Siamese Neural Network	[64]	Scopus	Conference publication	NQ	26	2018	United States

32	Efficient Detection of Phishing Attacks with Hybrid Neural Networks	[65]	Scopus	Conference publication	NQ	19	2019	China
33	Image-Based Phishing Detection Using Transfer Learning	[20]	Scopus	Conference publication	NQ	20	2019	Thailand
34	DeepAnti-PhishNet: Applying deep neural networks for phishing email detection	[66]	Scopus	Conference publication	NQ	50	2018	India
35	Detecting Malicious URLs Using a Deep Learning Approach Based on Stacked Denoising Autoencoder	[67]	Scopus	Book Series	NQ	32	2019	China
36	Deep learning based-phishing attack detection	[68]	Scopus	Conference publication	NQ	10	2019	India
37	Phishing URL detection via CNN and attention-based hierarchical RNN	[69]	Scopus	Conference publication	NQ	30	2019	China
38	A cognitive support for identifying phishing websites using bi-LSTM and RNN	[70]	Scopus	Conference publication	NQ	15	2019	India
39	NeuralAS: Deep Word-Based Spoofed URLs Detection against Strong Similar Samples	[71]	Scopus	Conference publication	NQ	27	2019	China
40	Detecting phishing websites through deep reinforcement learning	[72]	Scopus	Conference publication	NQ	20	2019	United States
41	Leverage temporal convolutional network for the representation learning of URLs	[73]	Scopus	Conference publication	NQ	20	2019	China
42	Deep learning framework for cyber threat situational awareness based on email and URL data analysis	[74]	Scopus	Book Series	NQ	53	2019	India

43	TypoWriter: A tool to prevent typosquatting	[75]	IEEE Xplore	Conference publication	NQ	30	2019	Bangladesh
44	Detecting Phishing Web sites and Targets Based on URLs and Web page Links	[76]	IEEE Xplore	Conference publication	NQ	20	2018	China
45	Deep Q-Learning and Particle Swarm Optimization for Bot Detection in Online Social Networks	[77]	IEEE Xplore	Conference publication	NQ	19	2019	India

Publications by Scientific Database After scrutiny of the selected publications, Fig. 3.2 illustrates the number of publications by scientific database; as can be observed, the most significant number of primary studies (23) were found in the Scopus database, followed by six publications found in Web of Science, six articles found in IEEExplore, and four in SpringerLink and ACM Library. In Taylor & Francis, one publication was found, but these were already included within Web of Science.

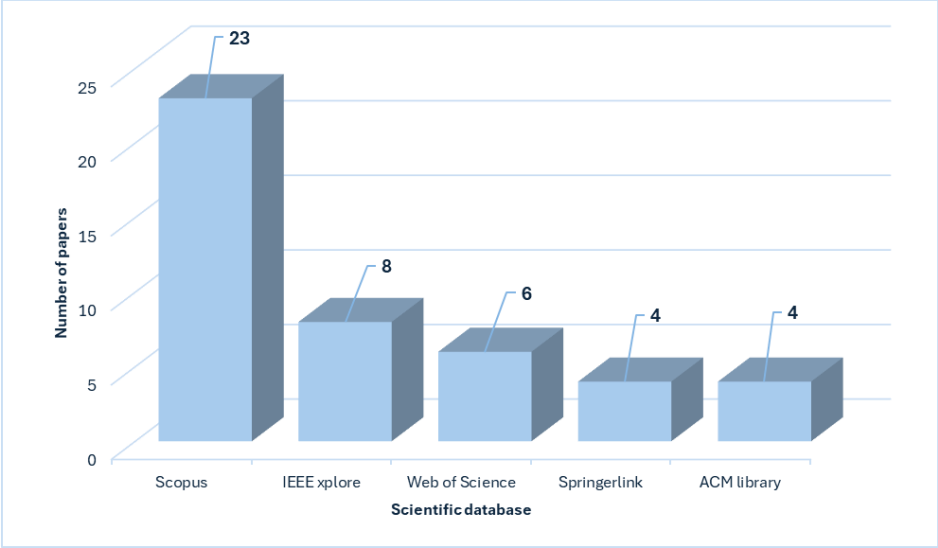


Figure 3.2: Number of publications by scientific database

Publications by Type The number of primary studies for each publication type is illustrated in Fig. 3.3. As can be observed, most of the researchers' contributions to this topic have been published in conference proceedings.

References by Publications The level of significant contribution has been analyzed to

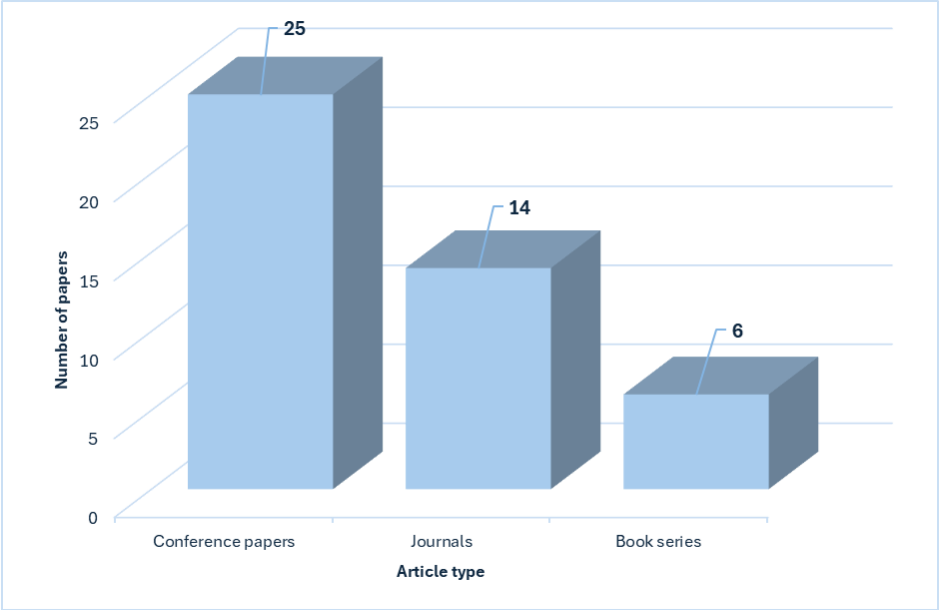


Figure 3.3: Number of publications by publication type

ensure each publication's quality and depth. This factor is about the number of references cited in each primary study; this can be seen in Table 3.1, in the TOTAL REFS column. For instance, the 51 references of the publication [25] is considerably higher than most of the primary documents.

Publications by Year The analysis of Phishing solution proposals using Deep Learning was considered since 2017, where three documents were found (see Fig. 3.4); in 2018, 21 documents were found; in 2019, 20 documents were found, and one document was found in 2020. Although Deep Learning is not new, its use has increased in recent years because two fundamental facts: (i) an enormous amount of data to analyze and (ii) more exceptional hardware capabilities. Its use, initially oriented to recognizing images and sounds, now extends to various fields, such as the early detection of Phishing attacks.

Publications by Country Figure 3.5 illustrates the countries that have contributed considerably to solving Phishing problems through Deep Learning. As can be observed, China has published 14 publications on the topic, India 12, followed by the United States with four, Slovenia and Colombia with two, respectively.

3.1.2. Analysis of Primary Studies

Some schemes proposed for automated removal and related vulnerability analysis techniques are discussed briefly in Table 3.2.

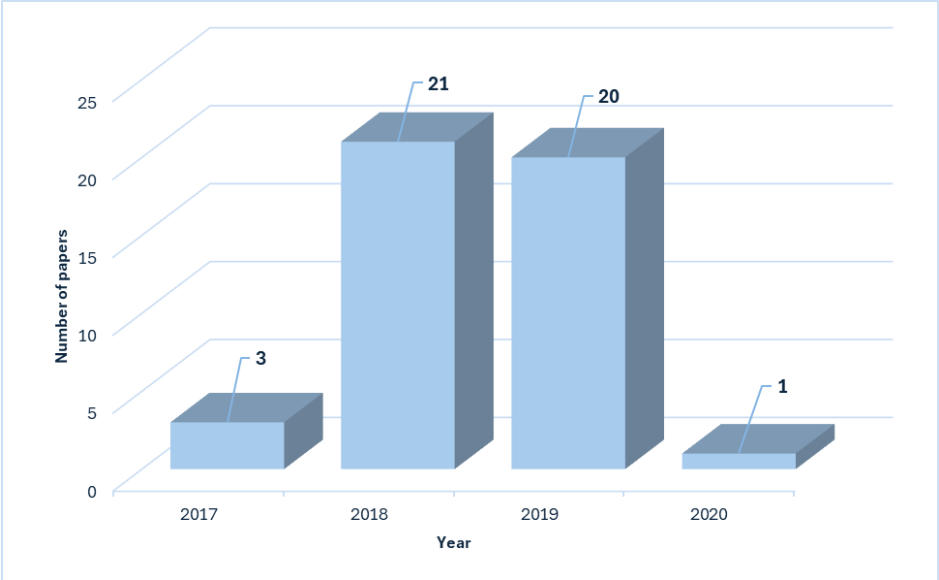


Figure 3.4: Number of publications by publication year

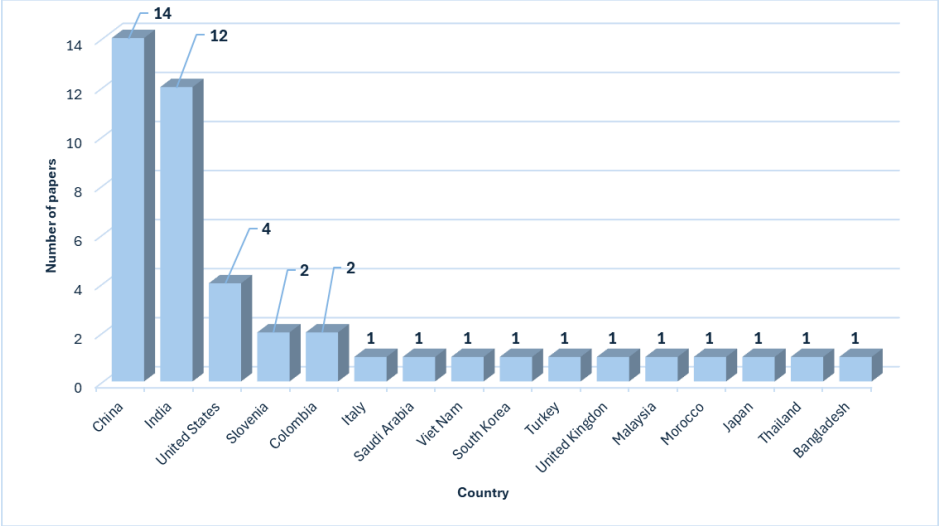


Figure 3.5: Number of publications by country

Table 3.2: Main contribution of each primary study.

Ref.	Phishing objective	Applied technique	Main features
[19]	E-mail	Hiransha et al. compare and reveal how to distinguish Phishing e-mails from legitimate e-mails. The dataset used had two types of e-mail texts, one with a header and the other without one.	This study evaluates the accuracy obtained with different epochs.

[20]	Other	Transfer learning is applied to classify whether the page is similar to a legitimate page. The novel image augmentation, which consists of sub-image identification and sub-image random placement, is proposed to prepare data for training the pre-trained CNN models.	The result revealed that the proposed method works effectively when a Phishing attack fakes a login page by scaling, moving, or removing some components.
[24]	Website	Wei et al. proposed a multi-spatial convolutional neural network to enable an accurate and efficient Phishing detection sensor. Extensive evaluations were conducted to show the performance of the proposed method. The true detection rate of the proposed method can achieve 86.63%.	A Raspberry Pi prototype was also implemented to enable real-time Phishing URL detection. With the proposed method, the execution time is reduced by 30%, and real-time detection is realized in a resource-constrained device.
[25]	E-mail	Fang et al. first analyzed the structure of the e-mail, which is divided into two parts, header, and body, and then went through multilevel embedding. Next, based on an improved recurrent CNN model with multilevel vectors and attention mechanisms, the authors propose a Phishing e-mail detection model named THEMIS.	THEMIS is used to simultaneously model e-mails at the e-mail header, the e-mail body, the character level, and the word level.
[30]	Website	A critical comparative evaluation is carried out. In particular, three techniques are evaluated: Traditional ML techniques, traditional ML techniques with Feature Selection, and recurrent neural networks (RNN).	This study determined that RNN is the best algorithm with 98.7% over Random Forest and Random Forest Classifier with Feature Selection.
[34]	Website	Selvaganapathy et al. proposed a methodology for detecting and categorizing malicious URLs using stacked restricted Boltzmann machines for feature selection with deep neural networks for binary classification.	The results demonstrate that Deep Learning-based feature selection and classification techniques can train the network quickly and detect reduced false positives.
[39]	Website	The authors propose a multidimensional feature Phishing detection approach based on a Deep Learning method. Finally, a quick classification result of Deep Learning into multidimensional features.	First, character sequence features of the given URL are extracted and used for quick classification by Deep Learning, and this step does not require third-party assistance or any prior knowledge about Phishing. Next, they combine URL statistics, web page code, and text features.

[40]	Website	The authors compared long short-term memory (LSTM) and the hybrid approach of LSTM with CNN.	The hybrid approach produces a better result than the pure LSTM approach, with 99.95% accuracy. This proves the hybrid approach is more accurate but very close to pure LSTM.
[41]	Other / Images	The authors intend to research whether Deep Learning methodologies for image classification can enable more practical and effective solutions. The main aim is Deep Learning, which recognizes false images in web browsers.	Experimental evaluation of a convolutional neural network resulted in high ranking accuracy for specific sets of 15 websites.
[42]	Website	This publication aims to use Borderline-Smote to solve the imbalanced data problem in the training of Phishing detection.	In this work, three features of a web page are used: URL features, page features, and image features.
[43]	Website	Wang et al. propose a model called PDRCNN, ensuring that this RNN- and CNN-based algorithm can detect the URL of a Phishing Website without relying on third-party data and search engines.	The model is very accurate compared to others. However, the disadvantage is its training time.
[45]	Website	The publication of Ali and Ahmed suggests hybrid intelligent Phishing website prediction approaches, which primarily focus on applying the enormous potential of DNNs together with consideration of genetic algorithm (GA) based feature selection and weighting methods to the problem of predicting Phishing websites.	Hence, feature selection can contribute reducing the number of features and eliminating irrelevant and redundant features.
[46]	E-mail	Nguyen et al. present a framework with hierarchical LSTM and attention mechanisms to simultaneously model the e-mails at the word and sentence levels. Also, it optimizes the training method of the model in combination with the characteristics of RNN.	The results show that this model reaches an accuracy of 99.1%, which is higher than that of other models of neural network algorithms.
[47]	Website	The authors analyze text messages on social networks to propose a Phishing URL recognition model based on neural networks and Deep Learning.	For this, a layered analysis is proposed, first using blacklists, second, shallow ML, and finally, Deep Learning LSTM.
[48]	E-mail	Vinayakumar et al. aim to show the abilities of word embedding to solve issues related to cybersecurity use cases. This work demonstrates the possibilities of amalgamating techniques from text analytics and Deep Learning for cybersecurity use cases.	Furthermore, this work attempts to use a CNN/RNN/multi-layer perceptron (MLP) network with Word2vec embedding on Phishing e-mail corpus, where Word2vec helps capture the synaptic and semantic similarity of Phishing and legitimate e-mails in an e-mail corpus.

[49]	Other	Hong et al. aimed to discover smishing attacks. To do this, images and texts of the messages are evaluated together and then passed on to a CNN. Hence, users with many posts are classified as certified users, while those with few posts are classified as unclassified.	A smishing attack is a phishing attack on chat messages from social networks, called social network services (SNS) messages. An important thing to highlight in this work is that it is not about finding the wrong Phishing users but rather guaranteeing the right users, according to their interests, based on the analysis of text and images of the users' SNS.
[50]	Website/ Images	Yi et al., use a metric learned in strings rendered as images instead of taking advantage of the similarity based on swapping and deleting characters in the URL address.	Later, CNN learns features optimized to detect the visual similarity of the rendered strings.
[51]	Website	Aksu et al. compared the following algorithms: feed-forward neural networks, stacked auto-encoders, support vector machines, and decision trees.	It is concluded that stacked autoencoders gave the best result among them. The accuracy percentage obtained (84%) is low, but this is related to the amount of small data entered, which was 2000.
[52]	Website	Chen et al. designed a detection system for Phishing websites using LSTM RNN.	Confusion of URLs is used to identify the different suspicious parts of the URLs.
[53]	E-mail	This research used two types of neural networks: the Feedforward Set Neural Network (EFFNN) and the Deep Learning Neural Network (DLNN).	The results showed that EFFNN is slightly better than DLNN.
[54]	Website	The authors address the parameter setting problem for a deep neural network utilizing swarm intelligence algorithms. In these experiments, authors applied the proposed method variants to the classification task for distinguishing between Phishing and legitimate websites.	The predictive performance of the resulting deep neural networks, trained using the parameter values as optimized with the help of the proposed swarm intelligence-based methods, improved significantly compared to the manually tuned neural network.
[55]	Website	A dual filter proposes a phishing detection architecture: one is for analyzing the textual content of the e-mail, and the second is for analyzing the suspect URL.	The algorithm proposed in this publication aims to accelerate the training speed by merely corrupting the URL characteristics through Auto-encoder and then reconstructing those characteristics through Denoise Auto-encoder.
[56]	Other	The authors propose a method for identifying malicious use of web certificates using deep neural networks. This system uses the content of TLS certificates to successfully identify legitimate certificates and malicious patterns used by attackers.	An algorithm is proposed to detect when a Transport Layer Security (TLS) protocol has been violated.

[57]	Website	The results of utilizing deep neural networks heavily depend on setting different learning parameters. To improve this, a swarm intelligence-based approach is proposed for the parameter setting of the Deep Learning neural network.	Applying the proposed approach to classifying Phishing websites makes it possible to improve their detection compared to existing algorithms.
[58]	E-mail	It is one of the few aimed at counterattacking spammers. For this, after the end-user reports possible spam using a plugin, authors' servers create a large group of URLs, employing chatbots, which respond automatically to the spammers.	The objective of the counterattack is to disable the services of the attackers.
[59]	Website	A tool is proposed to be deployed as a Chrome extension to overcome the problem of malicious URLs victimizing users. This tool analyses URLs and classifies them using two different neural networks, Artificial Neural Networks (ANN) and LSTM networks, a specific type of RNN. Later, it was observed that the training time is considerably less than that of the existing systems.	It is observed that the training time is considerably less than that of the existing systems. It can easily be deployed to devices with varied hardware specifications.
[60]	Website	In this work, a detection method based on LSTM was implemented. This study aims to solve the difficulty of other methods to extract the best features from the data.	Using the bag of bytes technique, a mechanical approach is applied to generate feature vectors from URL strings. This publication demonstrates that this prediction method is valid and can solve problems that traditional methods cannot.
[61]	Website	It presents a comparative study among traditional ML techniques with logistic regression using bigram, Deep Learning techniques like CNN, and CNN long short-term memory (CNN-LSTM) as architectures used to detect malicious URLs.	On comparison, CNN-LSTM gave the best accuracy of about 98%. In this publication, the authors claim that character-level URL embedding with Deep Learning layers can be a powerful method for automatic feature extraction.
[62]	Website	The solution proposed by Yao et al. is based on the evaluation of the original logos of the web pages through Deep Learning. The legitimacy of the URL in the two-dimensional code is evaluated using a legal logo embedded in the two-dimensional code.	This publication uses the Faster R-CNN method for small-scale identification and measures its impact on the FlickrLogos-32 dataset.

[63]	Other	Spaulding et al. present a system called D-FENS (DNS Filtering & Extraction Network System), which works in tandem with blacklists and features a live DNS server and binary classifier to predict unreported malicious domain names accurately.	The D-FENS classifier model operates at the character level and leverages Deep Learning architectures such as CNN and LSTM for real-time classification, which forgoes the need for feature engineering typically associated with traditional ML approaches.
[64]	Website/ Images	This publication presents a fundamentally different solution to homoglyph (name spoofing) attack, this problem using a Siamese convolutional neural network (CNN). This technique uses strings rendered as images.	The trained model converts thousands of potentially targeted processor domain names to feature vectors. This technique outperforms similar works, improving its detection by 13% to 15%.
[65]	Website	Zhang et al. pass the URL strings through Autoencoder and then through CNN. Autoencoder is adopted to reconstruct features that explicitly enhance the correlation relationship among the features.	It is concluded that a hybrid approach is more accurate than a traditional approach.
[66]	E-mail	Vinayakumar et al. use word embedding and Neural Bag-of-grams with Deep Learning methods such as CNN, RNN, LSTM, and MLP to detect Phishing e-mail, i.e., word embedding is evaluated with each technique.	It concludes that word embedding with Deep Learning, specifically LSTM, is appropriate for the anti-phishing task.
[67]	Website	Yan et al. employed a stacked denoising auto-encoder (SDA) network to analyze URLs and extract features automatically.	Logistic regression is implemented to detect malicious and benign URLs, generating detection models without manual feature engineering.
[68]	Website	Sumathi and Sujatha first obtained the most popular datasets, such as Ham, Phishing Corpus, and Phishload. Then, 30 features are obtained from these datasets to run Traditional ML and Deep Learning algorithms.	Finally, it is verified that the Deep Learning technique is much more accurate than ML.
[69]	Website	Huang et al. propose using a hybrid approach framework, which uses a detection model based on the CNN character level and a detection model based on the RNN word level.	This fusion results in a significant improvement in the accuracy of Phishing detection.

[70]	Website	This research proposes an intellectual structure that uses information highlights joined with semantic content and picture highlights to identify Phishing sites. The structure utilizes the ground-breaking profound learning system of Bidirectional LSTM RNN for Phishing recognition and consolidates it with Convolution Networks (CNN) for semantic picture highlight extraction.	The framework can be rediscovered from recently identified websites and tracked in a query database. Moreover, the area learning put away in the lookup database will help configuration designers recognize new Phishing systems as they develop.
[71]	Website	They propose Neural Anti-Spoofing (NeuralAS), in which contextual information within URL word sequences is used to detect malicious URLs.	The main idea is to segment URLs into sequences of words and use two-way RNN to carry out the detection with high abstraction characteristics.
[72]	Website	Chatterjee and Namin introduce a reinforcement learning (RL) based framework for automated URL-based Phishing detection.	This Deep Learning implementation of the RL algorithm is a complementary approach to the existing Phishing detection methodologies to make the system dynamic.
[73]	Website	The authors propose URL2vec to extract URLs' structural and lexical features and apply the temporal convolutional network for the URL classification task.	They compare the proposed solution with CNN, LSTM, and a hybrid CLSTM.
[74]	Website	The authors propose Deep-Spam-Phish-Net (DSPN), a Deep Learning-based Spam and Phishing detection framework. This framework contains two sub-modules. The first submodule detects Spam and Phishing Emails, and the second submodule detects Spam and Phishing URLs.	The framework can collect many security logs and extract optimal features to distinguish between benign and malignant activities.
[75]	Website	Regular users often mistype the URL of a web page (e.g., gooogle). This error is known as Typosquatting. Thus, Ahmad et al. propose the use of a tool called TypoWriter.	To create this tool, a set of n-grams is first generated (from original web pages), which are then used to train and test them using an RNN algorithm.
[76]	Website	It is proposed that a different approach based on Deep Learning be used, called Deep Forest. For their study, authors first select two kinds of features: URL-based and web page-based.	Next, these features are entered into different ML traditional algorithms and the Deep Forest algorithm for execution.

[77]	Website	The idea of using Particle Swarm Optimization (PSO), is to speed up the update of the agents installed in the client computers, by external bots, and thus, speed up the early detection of new Phishing attacks of Social Bots.	Social Bot is a software program that is widely used on Twitter, the social network, to generate fake accounts, spam URLs, and spam emails. For this reason, Lingam et al. propose to use an algorithm based on Deep Learning called DQL-PSO.

3.1.3. Results and Discussion

The following section summarizes the Deep Learning techniques and sub-techniques applied in the selected publications. Then, we analyze the phishing attacks handled by these techniques and sub-techniques identified before. Also, we analyze the amount of data used in each study and the accuracy of the proposed solutions. We summarize the data repositories used by publications (benign and malign web pages) that would be useful for upcoming studies. Finally, we present the validity of the research that was conducted.

Synthesis of Techniques and Sub-techniques Applied by Publication

The Deep Learning techniques applied in each publication are detailed in Table 3.3. The technique most used in primary studies was CNN, which was used 22 times, followed by RNN, which was used 20 times. On the other hand, SPN was not used. In addition to the Deep Learning techniques applied in Table 3.3, and considering that these techniques may have several sub-techniques, it is essential to know which sub-techniques were used. This can also be seen in Table 3.3. Also, it is highlighted that most primary studies used the sub-technique LSTM. Sub-techniques ESN, DBM, RMB, GAN, and DRBM are not shown in Table 3.3 because no primary studies were using these sub-techniques.

Table 3.3: Techniques and sub-techniques of Deep Learning applied by article.

publication	Techniques					Sub-Techniques						Data	Accuracy
	AE	RNN	BM	DNN	CNN	SAE	DAE	SDAE	LSTM	DBN	ReNN	(Records)	
[19]					X							4.583	96.80%
[20]					X							ND	ND
[24]				X	X							1.523.966	86.63%
[25]		X			X				X		X	8.787	99.85%
[30]		X							X			2.000.000	98.70%
[34]			X	X						X		ND	ND
[39]		X			X				X			2.010.779	98.99%
[40]		X			X				X			141.8	99.70%
[41]					X							1.500	99.00%
[42]										X		2.200	96.50%
[43]		X			X				X			500	97.00%
[44]					X							ND	95.50%
[45]				X								ND	ND
[46]		X							X			ND	ND
[47]									X			31.087	98.20%
[48]		X			X							18.778	95.20%
[49]					X							16.000	79.93%
[50]			X							X		ND	ND
[51]	X					X						ND	84.00%
[52]		X							X			4.000	99.14%
[53]		X							X			4.000	99.14%
[54]												159.710	96.65%
[55]	X						X					ND	ND
[56]		X							X			1.000.000	88.64%
[57]		X							X			ND	ND
[58]		X							X			ND	ND
[59]		X							X			2.000.000	96.89%
[60]					X							121.616	95.17%
[61]					X				X			ND	98.00%
[62]					X							ND	98.60%
[63]		X			X				X			ND	95.00%
[64]					X							ND	97.68%
[65]	X				X							ND	99.00%
[66]		X			X				X			18.778	99.10%
[67]								X				4.000.000	98.52%
[68]				X								15.509	92.00%
[69]		X			X							4.800.000	97.90%
[70]		X			X				X			ND	96.40%
[71]		X							X			347.949	98.40%
[72]				X								73.575	90.10%
[73]					X				X			417.851	95.97%
[74]		X			X				X			1.060.480	99.50%
[75]		X							X			223.5Gb	ND

[76]				X								6.197	93.98%
[77]												454.649	95.00%
TOTAL	3	20	2	6	22	1	1	1	21	3	1		

Synthesis of Phishing Attacks by Target

In our research, it has been verified that the two most common ways in which phishers attempt to attack users are through websites and emails. For this reason, it was possible to classify the primary studies according to attackers' target on websites, emails, and others (see Table 3.2). In the primary studies, proposals that applied a hybrid approach (i.e., they were oriented to solve problems of a web page, email, or combinations of them) were also found. As can be seen in Table 3.2, the most significant number of solution proposals focus on web page features (31), well above the proposals oriented to email features (7), leaving in last place proposals that use different approaches (7).

Phishing Web Pages The most common attack technique for conducting different types of attacks is through websites, specifically rogue websites [40]. Through a rogue website, an attacker can display content not requested by the user, and through deception, the attacker can steal personal information or cause financial fraud. Most documents focused on solutions to Phishing attacks using web pages, as seen in Table 3.2.

Phishing e-Mails The second most common technique for attacking end-users is through deceptive e-mails. The purest form of this type of attack is in which the phisher asks for the user's sensitive data in exchange for giving him some benefit. Another more elaborated type of Phishing e-mail attack is one in which a Phishing URL is included in the email body, thus seeking to redirect the user to a Phishing website (see Table 3.2). Our study found that nowadays, practically all works that aim to fight e-mail Phishing use e-mail to hook and redirect users to a Phishing website.

Other Techniques Table 3.2 indicates other techniques besides Phishing web pages and Phishing emails to face Phishing attacks. In the past, most Phishing attacks were made purely by email, such as an email in which users received a message that they had won a lottery prize and that the user had to send confidential information in response to the email to receive their prize. Now, users familiar with online payment methods and transactions avoid sending confidential data in emails. However, they are more confident about sending them through a web page because they are familiar with these electronic transactions. On the other hand, online programs can now make an identical copy of an original web page in

seconds. In this way, the primary role that emails play today in Phishing attacks is to offer a hook through a hyperlink to a Phishing website, making the user believe it is the link to a legitimate page.

Data and Accuracy by Publication

Table 3.3 depicts that the data used in each publication is variable. It ranges from 1.500 to 4.500.000 records. However, it is known that while more data is used for training in the application of Deep Learning techniques, the precision in detecting a threat increases.

As a premise, we know that Deep Learning algorithms produce more accurate results than traditional ML algorithms because the accuracy of Deep Learning algorithms increases depending on the amount of data entered to train the algorithm. These data entered in Deep Learning are commonly six digits high. That is why more credit should be given to the analyzed works that meet at least two conditions: (i) they exceed 98% accuracy, and (ii) their algorithms have been fed with at least 500,000 records. As a result, we note that the publications that meet these two properties are [30], [39], [43], [59], [67]. In particular, the solutions presented in [39], [43] are hybrids, proving that hybrid approaches always produce greater accuracy.

Data Repositories Used by Publication

For both training and testing Deep Learning techniques in the early detection of Phishing attacks, a large amount of data is commonly needed, in the order of 10^6 records. These data must be from both, malign web pages/emails and benign web pages/emails. Table 3.4 indicates the repositories of data used in each publication. The documents that are not included were those that did not specify which repository they used for their study.

The most used repository for collecting Phishing URLs was Phishtank, used in 23 of the 45 studies, while the most used for finding legitimate pages was Alexa in 13 of the 45 studies. Even propietary repositories can be created to collect phishing URLs or emails. However, this is impractical due to the large number of records required to train and test the Deep Learning algorithms.

Table 3.4: Data repositories used by each publication.

Ref.	Malign web pages							Benign web pages						Total data
	Phishtank [78]	Openphish [79]	Malwaredomainlist [80]	Phishload [81]	Phishlabs [82]	MalwareURL [83]	Millersmiles archive [84]	Alexa [85]	DMOZ [86]	UCI ML Repository [87]	Commoncrawl [88]	360Navigaton pages [89]	Starting point [90] /Yahoo [91]	(Records)
[20]	X													ND
[24]	X		X					X						1,523,966
[25]					X									8,787
[30]	X										X			2,000,000
[34]										X				ND
[39]	X								X					2,010,779
[40]	X	X	X			X		X	X					141,800
[41]								X						1,500
[42]	X											X		2,200
[43]	X							X						500,000
[45]	X						X			X			X	ND
[49]									X					16,000
[51]	X													ND
[52]								X						4,000
[53]								X						4,000
[54]	X						X	X		X			X	159,710
[55]								X						ND
[56]										X				1,000,000
[57]	X									X				ND
[58]	X													ND
[59]	X										X			2,000,000
[60]	X													121,616
[61]	X	X	X											ND
[63]	X	X	X					X	X					ND
[65]	X			X					X	X				ND
[67]	X							X			X			4,000,000
[68]				X										15,509
[69]	X	X						X						4,800,000
[71]	X							X						347,949
[72]	X													73,575
[74]	X	X	X					X	X					1,060,480
[76]	X													6,197
Total	22	5	5	2	1	1	2	13	6	6	3	1	2	

3.1.4. Open Challenges and Research Directions

Creating a framework that initially allows detecting if an attack is carried out utilizing web pages, e-mails, or another approach is encouraged; after doing so, collect the best characteristics, depending on the kind of threat, and deliver these characteristics to the Deep Learning algorithm that better fits its detection. Multiple studies have been conducted to detect Phishing, mainly on personal computers and servers; however, currently, the most used devices are mobile phones running Android or Apple operating systems. Therefore, other authors are encouraged to make a tool that fits these devices' hardware and software characteristics.

It would be essential to make an algorithm that adopts the best of the three main techniques of Phishing detection, that is, first, to evaluate if a threat is in a blacklist, second, to evaluate that threat employing a traditional ML algorithm, and finally, if it is not yet detected, to determine if that threat exists through a Deep Learning algorithm. Although deep learning surpasses the accuracy of traditional ML algorithms, this technique succumbs to the training required for this algorithm. Thus, future work is planned to make a model that accelerates the Deep Learning algorithms' training time. We also propose implementing an application to detect and warn the user, almost precisely and quickly simultaneously, if a page is phishing or not.

Commonly, phishers use domain names generated by Domain Generation Algorithms (DGAs), but to date, these domains have already been detected by different early detection tools. For this reason, Peck et al. [92] propose Charbot, a tool that allows the creation of fraudulent domain names without them being detected. We encourage researchers to develop a novel tool from domain names generated by d-grams, which will train a CNN algorithm that can combat these attacks.

We did not find any work that used deep learning and NLP simultaneously to detect phishing attacks. At that point, we set out to fill this gap in the research, creating a model that not only allows data to be entered into deep learning algorithms but also exploits the syntax and semantics of that data.

3.2. Literature Review of Phishing, Deep Learning, and NLP

Before conducting a study on the primary articles, we first conducted a meta-review of SLRs, literature reviews, and surveys from the last three years in the general field of Phishing

detection and Deep Learning to identify whether any studies have been identified focusing on the use of NLP to analyze the textual content of suspicious web pages.

The authors of the SLR [93] conducted an in-depth study on 100 articles to mitigate Phishing email attacks using NLP mixed with traditional ML or DL algorithms. The study describes that a high percentage of 38% of articles are oriented toward using the email body to detect attacks. It is striking that there is not a relevant percentage of detection of Phishing attacks based on the content of web pages.

Survey [94] comprehensively reviews all aspects surrounding Phishing attacks and their countermeasures, including DL techniques. It even references three solutions [95]–[97] that use text analysis of web pages to detect attacks. However, these solutions do not use NLP to detect Phishing attacks.

Next, once the search for information was conducted in secondary studies where we could not find articles proposing a solution similar to ours, we decided to do a more in-depth search, this time only on primary articles. To find articles related to this topic, we used the methodology proposed by Barbara Kitchenham [38] in a summarized way to perform an SLR. To follow this methodology, we first designed the research question; next, we explored the scientific databases in which we entered our search string based on the research question, applied the inclusion and exclusion criteria, applied the quality criteria, and finally, extracted the results.

3.2.1. Research Methodology

Research Question

Our main objective is to develop a model that allows us to detect Phishing attacks with high accuracy based on web pages' text content without wasting the text's intrinsic richness [98]. To achieve this, we used NLP; next, this dataset feeds a DL algorithm that classifies whether the data entered are from a Phishing or ham page (when a web page is benign or not Phishing). Based on this, the research question is defined as follows:

- Which primary studies give a solution to detect Phishing attacks using NLP and DL algorithms?

Scientific Databases

We perform the search in the following scientific databases:

- Elsevier;
- Scopus;
- Web of Science;
- IEEE Digital Xplore;
- ACM Digital Library;
- Springer Link.

Search String

This review aimed to find all primary studies on Phishing attack detection methods combining NLP and DL. Thus, the following string was entered into each scientific database:

> *Phishing and ("Natural Language Processing" or NLP) and ("Deep Learning" or LSTM or BiLSTM or GRU or BiGRU)*

Inclusion and Exclusion Criteria

The years we performed our search were 2017 to 2023 because this is when there has been the most movement in the DL algorithms and NLP exploitation. We only included articles in English. Furthermore, we only included primary articles that offer a concrete solution for detecting Phishing attacks. We searched for articles in Open Access Journals.

Quality Evaluation of Research

For the quality criteria of the search, it was determined that the review would only be performed in primary works that apply the analysis of the text contained in web pages, discarding other methods, such as URL analysis, because they were not our object of study. In addition, as described above, the review was performed on reputable scientific databases and considered only articles in Computer Science or similar.

3.2.2. Analysis of Primary Studies

Although the articles [99]–[103] of Table 3.5 implement DL and NLP techniques, all of them focus their research on analyzing only the content of the URL address of the web pages instead of the content of the web pages.

The article of [99] is not oriented toward detecting Phishing on web pages and is not based on detecting Phishing in the body of emails, but the approach to giving scores to the words is interesting. This score depends on two rankings obtained from these words, according to the ranking in which they appear in a Phishing email, Phishing Rank(w), or in legitimate emails, Legitimate Rank(w). For example, in this study, the following 20 words obtained the highest score: account 21.45%, it is 15.00%, click 14.11%, mailbox 9.59%, Cornell 9.58%, link 9.37%, verify 8.83%, customer 8.63%, access 8.50%, reserved 8.03%, dear 7.85%, log 7.70%, accounts 7.61%, paypal 7.52%, complete 7.37%, service 7.15%, protecting 6.95%, secure 6.94%, mail 6.70%, and clicking 6.63%. The hierarchical LSTM algorithm was used in this model.

The authors of [100] performed an algorithm using CNN self-attention. The proposed algorithm obtained an accuracy of 95.6%, even better than the accuracy obtained by the hybrid CNN-LSTM approach. In contrast to our proposal, this work did not analyze words, only URLs.

In [101], the authors used three steps to detect whether a page is Phishing. First, they extracted the lexical and host-based properties of a website. Second, they combined URL features, NLP, and host-based properties to train Machine Learning and DL models. NLP was only applied to the URL to detect if there was a similar word in that URL but not identical to a known domain. Furthermore, a single DL algorithm, the deep neural network, was used, resulting in an accuracy of 96.6%. The article did not analyze the content of the web pages. This study obtained an accuracy of 96.60%, although it did not indicate which DL algorithm was used.

The work of [102] is interesting because it included NLP and two DL algorithms (LSTM and CNN) to build a hybrid model to detect Phishing attacks. Still, in contrast to our work, this one is oriented toward something other than analyzing the web pages' contents but the URLs and images of the web pages.

An empirical study was performed in the model proposed by [104] to compare and determine which algorithm is better at detecting Phishing attacks: LSTM, CNN, or LSTM combined with CNN. It was determined that CNN was superior to the others in this experiment. On the other hand, the web pages' bodies were not analyzed, but the URLs of each website were. In this study, NLP was not used for pre-processing.

In [103], the application of NLP in detecting web pages was studied. For this, traditional ML and DL models were applied. The Deep Learning algorithms used were LSMT, GRU, and BiRNN. In this study, the analysis was performed only on URLs and not on the text of

the web pages.

In article [105], the authors proposed a model combining long-term recurrent convolutional and graph convolutional network algorithms to detect Phishing attacks on URL and HTML content. However, they did not use NLP.

An experiment to detect Phishing attacks using four DL algorithms like ours is [106]. The authors performed an intensive empirical study to determine which of the four algorithms detected a web page by its URL. They set multiple parameter values before running each DL algorithm. This article did not use NLP nor analyze the web pages' textual content. It was concluded that no algorithm produced the best measures on all performance metrics.

3.2.3. Results and Discussion

After applying the search string, we identified 28 articles (ten from Web of Science, nine from Scopus, five from Elsevier, and four from IEEE Explore), eliminating the identical results between the scientific databases, resulting in twenty articles. From the analysis of the 20 articles, the two main vectors of Phishing attacks that stood out were those carried out through Phishing emails or web pages. However, this research focuses on the text contained in the web pages; for this reason, 13 articles whose solutions were oriented toward Phishing emails were not considered. Finally, once the quality, inclusion, and exclusion criteria were applied, the seven articles presented in Table 3.5 were selected as the inputs for the present study. As we can see, published research in this common area still needs to be developed.

Table 3.5: Related research.

Article	**DL**	**NLP**	**Web Page Text**
[99]	Yes	Yes	No
[100]	Yes	Yes	No
[101]	Yes	Yes	No
[102]	Yes	Yes	No
[103]	Yes	Yes	No
[105]	Yes	No	Yes
[104]	Yes	No	No
Our model	Yes	Yes	Yes

3.2.4. Review of latest studies

At the end of our work, we conducted a final literature review to determine how this research will be positioned in the current state of the art. Thus, applying the research methodology, inclusion, exclusion, and quality criteria, the solutions similar to ours, which currently mitigate Phishing attacks through deep learning and NLP, are shown in Table 3.6.

Table 3.6: Latest Studies of Phishing, deep learning, and NLP.

Ref.	Title	Year
[107]	Towards a Hybrid Security Framework for Phishing Awareness Education and Defense	2024
[108]	Uncovering SMS spam in swahili text using deep learning approaches	2024
[109]	Life-long phishing attack detection using continual learning	2023
[110]	Cloud-based email phishing attack using machine and deep learning algorithm	2023
[111]	A Deep learning-based innovative technique for phishing detection in modern security with uniform resource locators	2023
[112]	Attention-based 1D CNN-BILSTM hybrid model enhanced with FastText word embedding for Korean voice phishing detection	2023
[113]	Hybrid features by combining visual and text information to improve spam filtering performance	2022
[114]	An effective detection approach for phishing websites using URL and HTML features	2022

The model in [107] combines phishing detection and the generation of phishing content for user education, using AI and deep learning techniques, which integrates the detection of phishing attacks and end-user education, using advanced language and deep learning architectures, improving defense effectiveness.

In [108], the model combines several deep learning architectures (CNN, LSTM, etc.) and categorizes SMS messages into spam and ham. It shows high accuracy, reducing the need for manual engineering of characteristics and allowing more autonomous learning, improving

efficiency and accuracy in classifying SMS messages.

In [113], the main features include a hybrid feature set and the use of an XGBoost classifier for phishing detection, which is improved by using a hybrid feature set that provides URL character sequences, hyperlink information, and features based on textual content, overcoming the limitations of previous URL-only approaches.

In [109], the main features include continuous learning, which allows the model to adapt to new threats without forgetting previous knowledge. This and advanced embedding techniques reduce performance drops in evolving phishing attack scenarios.

In [110], key features include feature extraction using NLP, multiple classification algorithms (SVM, NB, LSTM), and creating a modified dataset to improve phishing detection. These improvements improve accuracy and execution time compared to other ranking reports, providing a more robust ranking report.

In [111], the main features include the use of a combination of deep learning techniques (CNN and LSTM) and the consideration of multiple attributes of URLs to improve the accuracy of phishing detection, which is improved by considering user behavior and emerging threats, providing a more effective phishing detection strategy compared to previous models that only used textual features.

In [112], the main features include using a hybrid CNN and BiLSTM model, integrating FastText for embedding, and implementing an attention mechanism to improve generalization. This enhances the generalization and robustness of the model by helping to reduce overfitting compared to previous models that might have had extraordinary performance due to this issue.

In [113], it includes using three sub-models to extract text and image features and a classifier model that combines these features to improve spam detection. Combining text and image features allows for more accurate classification of spam images, overcoming the limitations of previous methods that only considered text.

Only four of the articles in Table 3.6 [107], [109], [111], and [114] focus on detecting phishing attacks on web pages, and just one [114] detect attacks in the content of web pages like ours. On the other hand, articles [109], [110], and [113] focus on detecting these attacks on emails. The model of article [109] is hybrid; it applies to web pages and emails. The articles [108] and [112] select different approaches. In [108], the detection is made on the SMS messages; in [112], the detection is carried out on the voice of an interlocutor on the telephone line.

Article [114], the most related to our proposal, proposes a phishing attack detection solu-

tion based on several features, including the HTML content of a web page using embedding with the GloVe dictionary. The main difference between our proposal and that of [114] is that in [114], the algorithm is applied to the HTML code and not only to the web page's text. That is why it is necessary to conduct a study that evaluates how much the accuracy varies between feeding an algorithm with HTML code and text. This study is carried out in Chapter 4.

3.3. Chapter Summary

In the first part of this chapter, a Literature Review was carried out to understand how deep learning is used to combat phishing attacks. In the second part, a Literature Review was conducted to understand and locate our research on mitigating attacks with NLP and deep learning.

To carry out the first literature review, information was searched in five scientific digital databases, which resulted in 180 articles found initially. From these publications, once the inclusion and exclusion criteria were applied, only 45 primary studies were selected as the material for our research. It can also be noted that the most significant number of solutions is aimed at the largest number of attack vectors, the Phishing web pages, followed by Phishing e-mails.

In the 45 primary studies selected, several Deep Learning techniques are used to detect Phishing attacks; CNN leads the solutions; this technique is applied 22 times, followed by RNN, which is applied 20 times. On the other hand, the sub-technique LSTM of RNN is the sub-technique most used (21 times). To use the studied techniques, and as a characteristic of Deep Learning, it is necessary to have an enormous amount of data because the higher the amount of data, the higher the precision of the technique. To this end, it was determined that the site where most Phishing page information can be used is `phishtank.com`. On the other hand, to train the algorithms, an enormous number of benign pages is necessary; hence, the most commonly used site for these benign pages is `alexa.com`.

Among the reviewed datasets, `phishload.com` stands out because it contains the HTML code of Phishing and benign pages. For this reason, we selected this dataset that also fits as a supervised and binary algorithm to continue with our research.

The techniques most used in mitigating this type of attack are the Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN), among other findings. From now on, we will use Deep Learning algorithms such as RNN and CNN and some of their sub-

algorithms.

In this first part, we did not find studies that combined deep learning and NLP to mitigate Phishing attacks; therefore, this path was chosen because the better the data is purified, the better the model's accuracy.

In the second part of this chapter, another literature review was carried out to determine how our research is located in this rapidly growing discipline of artificial intelligence. Thus, our research is well-placed compared to the current state of the art. Only one article [114] is similar to the one in this thesis, which complies with phishing detection with deep learning and NLP in the text obtained from web pages using word embedding.

Chapter 4

Phishing Detection on HTML Code vs Text

Contents

As concluded in the literature review, several proposals use deep learning with HTML to detect phishing attacks, and almost none perform detection on text obtained from web pages. On the other hand, it is necessary to determine which deep learning algorithm (DNN, RNN, CNN, and RCNN) is the best suited to solve this type of detection problem. This is why this study is conducted to evaluate metrics obtained from HTML and text, combined with DNN, RNN, CNN, and RCNN.

4.1. HTML vs Text Hypothesis

It was hypothesized in this chapter that it does not matter whether deep learning algorithms are fed with HTML code or with text obtained from that HTML code; that is, the precision does not vary significantly between text and HTML code. This hypothesis arose from our observation of the detection of phishing attacks by email [6], in which a similar study is

carried out but focused on the body of the emails. In [6], the body of the email containing words such as won, money, and credit card, among others, was detected for detection.

The literature reviewed in chapter 3 contains several solutions that apply Deep Learning to HTML code or text obtained from that HTML code; however, the literature does not do a comparative study to detect phishing attacks between HTML codes and the text obtained from web pages. Therefore, this chapter presents a comparative analysis of deep learning algorithms to determine if it is more effective to detect an attack, either using HTML code or the text obtained from this code. Hence, the Deep Neural Network (DNN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Recurrent Convolutional Neural Network (RCNN) algorithms were executed, feeding them first with HTML code and then with text. The average of the metrics obtained with HTML was 85%, and the overall metrics obtained for text averaged 84%. In conclusion, it is determined with this study that it makes no difference whether the algorithm is fed with HTML or text because when analyzing with text, the unnecessary features of HTML are eliminated. Still, simultaneously, the essential elements of HTML are lost.

4.2. Methodology

The methodology used to make the study of this chapter is divided into three groups: data collection, data cleaning, and execution of the selected Deep Learning algorithms. Figure 4.1 shows the steps performed to make this study:

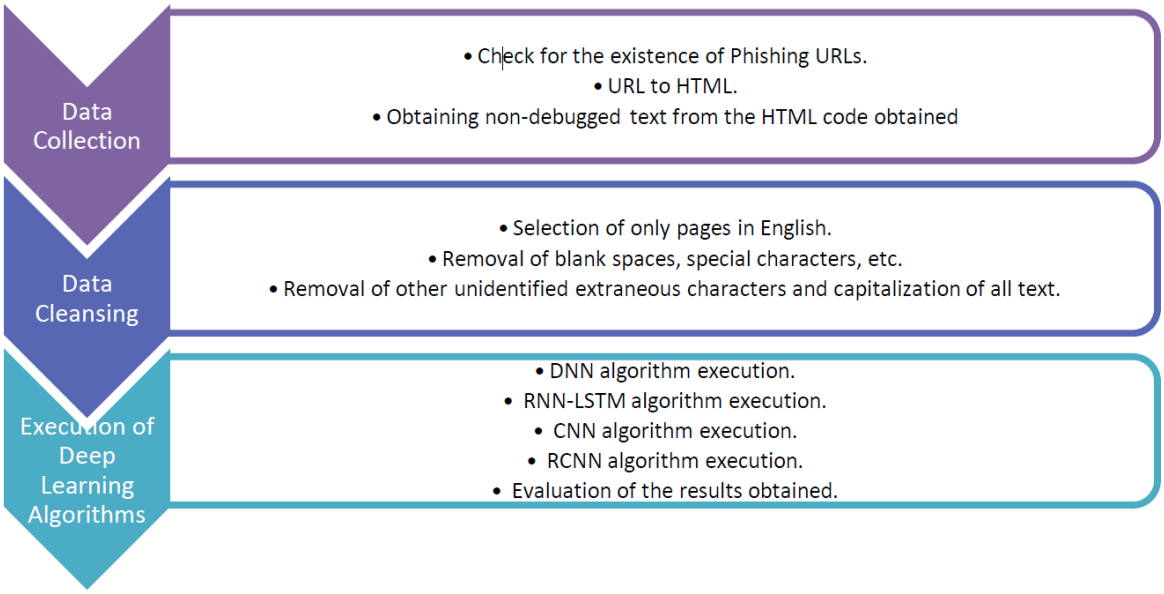


Figure 4.1: Research Methodology to Evaluate HTML code vs Text

4.2.1. Data Collection

Our study aimed to detect attacks on web pages using their content. Thus, we extracted the data from the three sources shown in Table 4.1.

Table 4.1: Phishing and Ham Datasets.

Dataset	Phishing	Ham	URL	HTML
Phishtank [78]	7,983	0	Yes	No
Phishload [115]	9,312	1,176	Yes	Yes
Malicious URL [116]	90,000	90,000	Yes	No

As our study in this chapter aimed at detecting attacks on web pages using only their content, it was necessary to implement algorithms to obtain the HTML code from the URL and, next, only the text from that HTML. The data was obtained from the three different sources shown in Table X, so it was necessary to convert the URL formats (domain.com) to a single ISO format (http://domain.com). Therefore, data in ISO format can now be parsed by Python. The procedure used can be seen in Figure 4.2.

Figure 4.2: Steps in data collection.

Check for the existence of URLs

One of the characteristics of the webpages created to carry out phishing attacks is that they disappear after a short time. For this reason, it was necessary to carry out an algorithm to check if these pages were still active, keep only those active websites, and delete the others. It should be noted that this process took the longest execution time on the server, even though this server is up to 60 times faster than a personal computer. Thus, one of the processes that could take up to 100 days on our personal computer was carried out on the server in only two days.

URL to HTML

Since the Phishtank and Malicious_URL records did not have their HTML content, it was necessary to create an algorithm to obtain this code. An example of this result is shown in Figure 4.3.

```
1 print(df1.at[1,'htmlContent'])
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<script type="text/javascript">var ue_t0=ue_t0||+new Date();</script>



<!--btech-iplc-->
   <script type="text/javascript">
     new Image().src="http://g-ecx.images-amazon.com/images/G/01/gno/images/orangeBlue/navPackedSprite
g";
     new Image().src="http://g-ecx.images-amazon.com/images/G/01/x-locale/common/transparent-pixel._V1
   </script>
    <meta content="on" http-equiv="x-dns-prefetch-control" />
    <link href="http://g-ecx.images-amazon.com" rel="dns-prefetch" />
    <link href="http://z-ecx.images-amazon.com" rel="dns-prefetch" />
    <link href="http://ecx.images-amazon.com" rel="dns-prefetch" />
```

Figure 4.3: Example of HTML code obtained from a webpage.

HTML to text

The text was extracted once each web page's HTML code was obtained. More than 12,000 records were obtained, of which about 6,000 are Phishing and 6,000 are Ham. This text obtained is not yet valid for input to the Deep Learning algorithms, so preprocessing is carried out in preprocessing.

4.2.2. Preprocessing

Up to this point, the text obtained from each web page has already been achieved; however, analyzing it with our Deep Learning algorithms still needs to be more helpful because they still contain pages in English with unwanted or non-alphabetic characters. Putting all the text in English is necessary, so we continue with the preprocessing steps described in Figure 4.4.

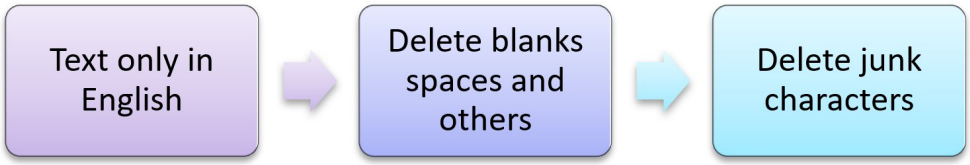


Figure 4.4: Steps in the preprocessing.

Text only in English

Our study is also based on detecting attacks utilizing the text on the webpage, so it was necessary to carry out an algorithm to keep only those records whose text is only in English. Only about 10,000 records remained between Phishing and Ham, thus maintaining the data balance between the two classes. See Figure 4.5 with a non-English page.

```
print(df1.at[0,'textBS'])
```

```
Google Wenn Sie ein Bildschirmleseprogramm verwenden, klicken Sie hier zum Deak
tivieren von Google Instant. +IchSucheBilderVideosMapsNewsShoppingMailMehrÃœber
setzerBÃ¼cherScholarBlogsYouTubeKalenderFotosDocsSitesGroupsReaderNoch mehr Â»A
ccount OptionsAnmeldenSucheinstellungenErweiterte SucheSprachoptioneniGoogleWeb
protokoll           Google                                          Google Instant
ist derzeit nicht verfÃ¼gbar. DrÃ¼cken Sie die Entertaste, um die Suche zu star
ten. Weitere Informationen  Google Instant ist aufgrund zu geringer Verbindungs
geschwindigkeit deaktiviert. DrÃ¼cken Sie zum Suchen die Eingabetaste. Zum Star
t der Suche Eingabetaste drÃ¼cken  Zum Start der Suche Eingabetaste drÃ¼cken
| Deutschland  Werben mit GoogleUnternehmensangeboteNeue DatenschutzerklÃ¤rung
und NutzungsbedingungenÃœber GoogleGoogle.com in EnglishiGoogleHintergrundbild
Ã¤ndern
```

Figure 4.5: Text record from a non-English page.

Delete junk characters

With the previous step, many characters not used in our algorithm were deleted; however, some junk characters could still affect performance. Also, in this step, the numbers in the text were deleted. So far, the text is clean, but in upper and lower case; however, it is necessary to transform all text to upper case to prevent the algorithms from treating the exact words differently, for example, play, Play, and PLAY. The upper function was used to transform all the text to the upper case. It's crucial to delete the Stop Words or empty words, which are words without meaning, such as articles, pronouns, prepositions, and others. However, we've decided strategically to filter out these words later when implementing the deep learning algorithms. This decision reflects our careful planning and foresight in the process. Until now, the text has already been clean and ready to be entered into deep learning algorithms. See Figure 4.6.

```
print(df1.at[1,'text_html_cleaned'])
```

```
AMAZONCOM ONLINE SHOPPING FOR ELECTRONICS APPAREL COMPUTERS BOOKS DVDS MORE A DIFFERENT VERSION OF TH
IS WEB SITE CONTAINING SIMILAR CONTENT OPTIMIZED FOR SCREEN READERS AND MOBILE DEVICES MAY BE FOUND A
T THE WEB ADDRESS WWWAMAZONCOMACCESS AMAZONCOM HELLO SIGN IN TO GET PERSONALIZED RECOMMENDATIONS NEW
CUSTOMER START HERE A FREE TWODAY SHIPPING SEE DETAILS YOUR AMAZONCOM A A A A A A TODAYS DEALS A A GI
FTS WISH LISTS A A GIFT CARDS A YOUR DIGITAL ITEMS A A YOUR ACCOUNT A A HELP SHOP ALL DEPARTMENTS SEA
RCH ALL DEPARTMENTSAMAZON INSTANT VIDEOAPPLIANCESAPPS FOR ANDROID ARTS CRAFTS SEWINGAUTOMOTIVEBABYBEA
UTYBOOKSCELL PHONES ACCESSORIESCLOTHING ACCESSORIESCOMPUTERSELECTRONICSGIFT CARDSGROCERY GOURMET FOOD
HEALTH PERSONAL CAREHOME KITCHENINDUSTRIAL SCIENTIFICJEWELRYKINDLE STOREMAGAZINE SUBSCRIPTIONSMOVIES
TVMP DOWNLOADSMUSICMUSICAL INSTRUMENTSOFFICE PRODUCTSPATIO LAWN GARDENPET SUPPLIESSHOESSOFTWARESPORTS
OUTDOORSTOOLS HOME IMPROVEMENTTOYS GAMESVIDEO GAMESWATCHES A A CART CART WISH LIST ALL LISTS REGISTRI
ES AEURO WISH LIST AEURO GIFT ORGANIZER AEURO WEDDING REGISTRY AEURO BABY REGISTRY AEURO AMAZON REMEM
BERS UNLIMITED INSTANT VIDEOS PRIME INSTANT VIDEOS UNLIMITED STREAMING OF THOUSANDS OFMOVIES AND TV S
HOWS WITH AMAZON PRIME LEARN MORE ABOUT AMAZON PRIME AMAZON INSTANT VIDEO STORE RENT OR BUY HIT MOVIE
S AND TV SHOWSTO STREAM OR DOWNLOAD YOUR VIDEO LIBRARY YOUR MOVIES AND TV SHOWSSTORED IN THE CLOUD WA
```

Figure 4.6: Example of a text record ready for input to deep learning algorithms.

4.3. Results of the execution of deep learning algorithms

Once the text was cleaned, the Deep Learning algorithms were run on this data. The columns that were entered for the execution of each of the algorithms were:

- html_Content: It has stored HTML code obtained from each URL.
- text_html_cleaned: It has stored all the clean and pre-processed text.
- isPhish: Indicates whether that record corresponds to a Phishing page.

The Deep Learning algorithms chosen to enter and execute the data were the following:

1. Deep Neural Network (DNN)
2. Recurrent Neural Network (RNN)
3. Convolutional Neural Network (CNN)
4. Recurrent Convolutional Neural Networks (RCNN)

These algorithms are similar to those used in the paper [117], in which a selection problem is solved with 20 classes. Our solution is focused on two classes (Phishing or Ham). First, each algorithm was run with HTML code and then with clean text to determine which of them, HTML or text, obtained the best results.

4.4. Comparison of Deep Learning Algorithms

Table 4.2 shows the values obtained when running each algorithm with HTML code. It can be seen that DNN, CNN, and RCNN give similar results of 86% in all their metrics, while RNN was the worst performer with 82%.

Table 4.2: Precision, recall, F1-score, and accuracy values obtained with each Deep Learning algorithm on HTML code.

	DNN	**RNN**	**CNN**	**RCNN**	**AVERAGE**
Precision HTML	0.86	0.82	0.86	0.86	0.85
Recall HTML	0.86	0.82	0.86	0.86	0.85
F1-score HTML	0.86	0.82	0.86	0.86	0.85
Accuracy HTML	0.86	0.82	0.86	0.86	0.85

Table 4.3 shows the values obtained when running each of the four algorithms with text only. DNN, RNN, CNN, and RCNN give similar results at 84%; however, CNN generally lowers the percentage, while RCNN raises the Precision metric.

Table 4.3: Precision, recall, F1-score, and accuracy values obtained with each Deep Learning algorithm on text.

	DNN	RNN	CNN	RCNN	AVERAGE
Precision Text	0.84	0.84	0.84	0.85	0.84
Recall Text	0.84	0.84	0.83	0.84	0.84
F1-score Text	0.84	0.84	0.83	0.84	0.84
Accuracy Text	0.84	0.84	0.83	0.84	0.84

Based on the averages obtained from 4.2 and 4.3, the best average is obtained with HTML, 85%, over text, 84%. In other words, it cannot be determined that when running the algorithms with either HTML or text, one is better than the other because the percentage difference of one point is slight. In general, when running the same algorithms with the same data, the variation of percentages is low.

4.5. Chapter Summary

Several investigations detect phishing attacks based on the content of web pages; However, most of these investigations use Deep Learning algorithms applied only to the HTML code of the web pages and not to the clear text obtained from the HTML code. For this reason, in this chapter, we presented the results of a comparative study to determine if it is more accurate to analyze the HTML code or the text obtained from that HTML code. We evaluated the accuracy between HTML and text input of four Deep Learning algorithms: Deep Neural Network (DNN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Recurrent Convolutional Neural Networks (RCNN).

When obtaining clear text from the HTML code, a trade-off is made between removing irrelevant features and losing relevant features in the process. While the text obtained removes many irrelevant features, it also results in the loss of some relevant ones. For instance, a deleted link that always points to an external Phishing page is a relevant feature that could potentially be lost. This link would be crucial for determining whether a page is Phishing. Therefore, from the results of the execution of the four algorithms, it can be concluded that whether we input HTML code or clear text obtained from that code into any Deep

Learning algorithm is irrelevant.

Nevertheless, one key benefit of using clean text instead of HTML code in training and testing Deep Learning algorithms is the significant reduction in database size. Our algorithm, which inputs a clean dataset with text, allows for faster training and testing; this is because the database size is reduced from 600 Mbytes in HTML code to 30 Mbytes in clean text, improving algorithm performance efficiency.

In this particular comparative study, the Deep Learning algorithms were fed directly with text obtained from the HTML code. In the following chapters, we present our proposed model developed to analyze the text semantically and syntactically before it is fed into the algorithms to improve the accuracy and precision of Phishing detection. In this way, we will take advantage of the richness of grammatical structures and the meaning of words.

Chapter 5

Materials and Methods

Contents

5.1. Proposed Model Based on NLP and DL

This research proposes a model based on NLP and DL to pre-analyze the text entered into the DL algorithm. For this purpose, our model comprises four phases from the HTML code: word parsing, data pre-processing, feature representation, and feature extraction, as illustrated in Figure 5.1.

Figure 5.1: Phishing attack detection—overview of the proposed model

Thus, with our method, first, by word parsing, the input text is divided into words, and the relationships between these words are obtained; then, by pre-processing with NLP, two phases are used: data pre-processing and feature representation, by which the most important words to be analyzed are obtained, in addition to the importance of the meaning of the order in which the words are entered. Finally, by DL, the essential features of the input text are obtained automatically, and the algorithm is trained. The final goal of the model is to obtain greater precision in detecting phishing attacks.

An essential contribution of our model is that before these data are entered into the DL algorithms, they go through a word embedding process, specifically Keras Embedding and GloVe. As an additional contribution, four DL algorithms were evaluated to determine which best fit our model.

The phishing detection problem is a binary classification task because the algorithm can only result in two options: phishing or ham. Thus, a dataset with 10,373 rows and two columns was obtained after processing. Therefore, the first column contains clean text, i.e., text without characters that negatively affect the accuracy of the experiment. The second column indicates whether that row is phishing or not. For practical purposes, phishing = 1 and ham = 0. Each of the stages and sub-stages of the model is described in the following subsections.

5.1.1. Word Parsing

From the Phishload database [115], we obtained 10,373 records of HTML code from phishing and ham pages. However, if we enter all that text to be analyzed by our selected DL algorithms, the results will be less accurate. For example, let us examine the following set of words found in the vast majority of HTML code of phishing pages: “DOCTYPE HTML PUBLIC”. The algorithms will detect that these features are an essential indicator to decide that a page is phishing when it is not. For this reason, in our model, we first cleaned the text using the Regular Expressions and four Natural Language Toolkit (NLTK) tools. In Figure 5.2, the reader can see all the steps followed in word parsing.

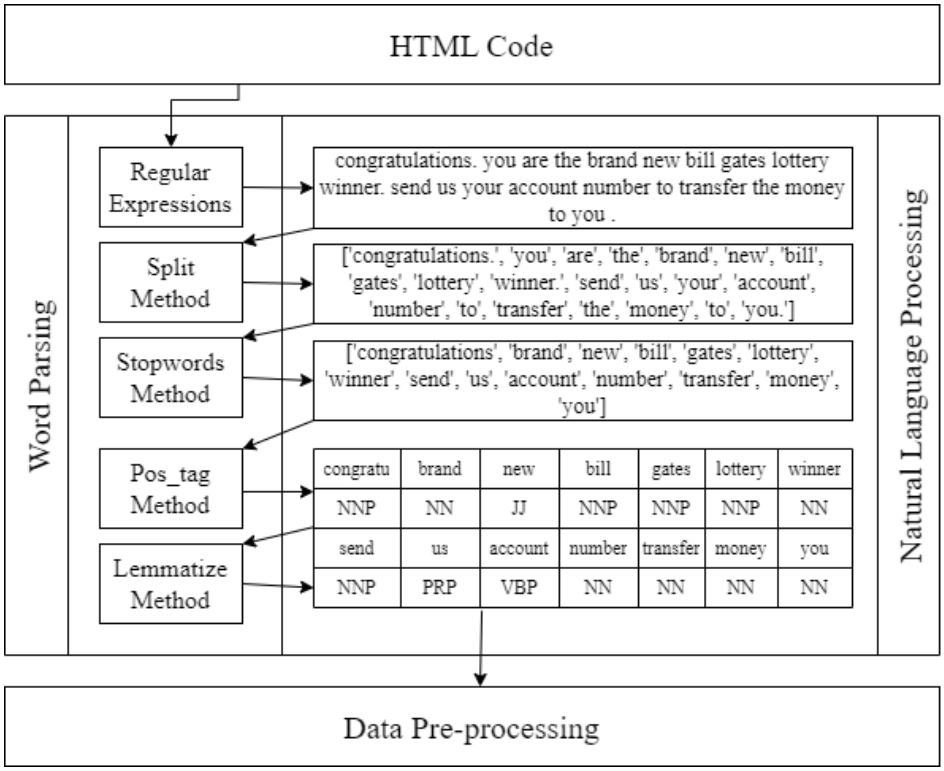


Figure 5.2: Word parsing sub-processes

Regular Expressions

Initially, the text obtained from the web pages still contains objects such as strings, numbers, characters, and so on, which are unnecessary for the analysis. Therefore, we first transformed all letters to lowercase to continue our analysis. Then, by regular expressions, we removed URL addresses, mentions with @ or #, HTML tags, digits, and all junk characters that need to be deleted.

Split Method

The split method is a string method that is used to split a string into a list of smaller substrings. The method takes a separator as an argument and divides the string based on the occurrence of that separator. If we execute the following code with a known phishing string:

text = "congratulations you are the brand new Bill Gates lottery winner send us your account number to transfer the money to you",
words = text.split()
print(words)

Then, the result will be as follows:

['congratulations', 'you', 'are', 'the', 'brand', 'new', 'bill', 'gates', 'lottery', 'winner', 'send', 'us', 'your', 'account', 'number', 'to', 'transfer', 'the', 'money', 'to', 'you']

Stopwords Method

Now, we cleaned the text of stopwords or empty words that do not have any meaning for our analysis because they can appear in phishing and ham text. For example, the words the, or, and, that, this, and of are deleted from the text.

Let us analyze the string obtained in the previous step with the following code:

```
stop_words = set(stopwords.words('english'))
words = ['congratulations', 'you', 'are', 'the', 'brand', 'new', 'bill', 'gates', 'lottery', 'winner', 'send', 'us', 'your', 'account', 'number', 'to', 'transfer', 'the', 'money', 'to', 'you']
filtered_words = [word for word in words if word.casefold() not in stop_words]
print(filtered_words)
```

When *stopwords Method* is executed on the example text, the following words are deleted:

'you', 'are', 'the', 'your' and *'to'*

Hence, the set of words that stays are:

['congratulations', 'brand', 'new', 'bill', 'gates', 'lottery', 'winner', 'send', 'us', 'account', 'number', 'transfer', 'money', 'you']

Pos_Tag Method

The pos_tag method is used for Parts-Of-Speech (POS) tagging words in a text. POS tagging is the process of assigning a part of speech, such as a noun, verb, adjective, and so on, to each word in a text. Let us execute the following commands on the string obtained in the previous step:

```
tokens = ['congratulations', 'brand', 'new', 'bill', 'gates', 'lottery', 'winner', 'send', 'us', 'account', 'number', 'transfer', 'money', 'you']
pos_tags = pos_tag(tokens)
print(pos_tags)
```

This results in the following duos:

[('congratulations', 'NNP'), ('brand', 'NN'), ('new', 'JJ'), ('bill', 'NNP'), ('gates', 'NNP'), ('lottery', 'NNP'), ('winner.', 'NN'), ('send', 'NNP'), ('us', 'PRP'), ('account', 'VBP'), ('number', 'NN'), ('transfer', 'NN'), ('money', 'NN'), ('you', 'NN')]

These duos indicate which part of each sentence constitutes each word. However, for our syntactic and semantic analysis, we only considered words that are nouns (start with N: NN, NNP), verbs (start with V: VBP), adjectives (start with J: JJ), and adverbs (start with R).

For this reason, the pair *('us', 'PRP')* was eliminated, leaving only the following pairs:

[('congratulations', 'NNP'), ('brand', 'NN'), ('new', 'JJ'), ('bill', 'NNP'), ('gates', 'NNP'), ('lottery', 'NNP'), ('winner.', 'NN'), ('send', 'NNP'), ('account', 'VBP'), ('number', 'NN'), ('transfer', 'NN'), ('money', 'NN'), ('you', 'NN')]

Lemmatize Method

The lemmatization process involves reducing words to their base or root form, which can be helpful in NLP. The base or root form of a word is called its lemma. The NLTK library provides a WordNetLemmatizer class that can be used for lemmatization. As an example, let us execute the following code:

```
words = ["cats", "running", "ate"]
lemmas = [lemmatizer.lemmatize(word, pos='v') for word in words]
print(lemmas)
```

Which produces the following output:

"cat", "run", "eat"

It can be observed that the words cats, running, and eating are lemmatized to their base forms "cat", "run", and "eat", respectively. By lemmatizing words, we can reduce the number of unique words in a text corpus, improving the accuracy and efficiency of many natural language processing tasks.

5.1.2. Data Pre-Processing

Once all the content of the web pages has been word parsed, it is still necessary to pre-process the data to obtain data that is ready to be entered into the feature representation

using Keras Embedding with GloVe. For this purpose, three consecutive steps were followed, as shown in Figure 5.3: tokenization, encoding, and padding.

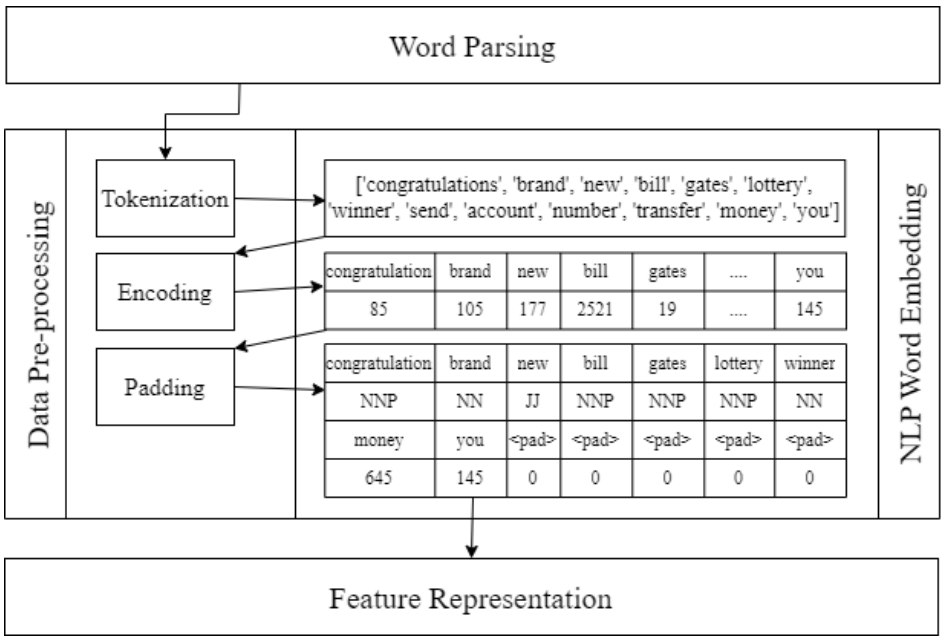


Figure 5.3: Data pre-processing sub-processes

Tokenization

Tokenization is a necessary process for any task involving NLP. It consists mainly of vectorizing each word of text obtained previously, resulting in a sequence of meaningful words. In our case, we used the texts_to_sequences tokenization method to transform the input string into a sequence of integers. Let us run the following tokenization commands on the above string example:

words = "congratulations brand new bill gates lottery winner send account number transfer money you"
tokens = word_tokenize(words)
print(tokens)

This split each word of the input string:

['congratulations', 'brand', 'new', 'bill', 'gates', 'lottery', 'winner', 'send', 'account', 'number', 'transfer', 'money', 'you']

Encoding

To encode the words in the text string, we used the texts_to_sequences function, which incrementally assigns an integer to each word as it appears. This way, we convert the text data into a numeric format that the DL models can process. There are other encoding techniques, such as one-hot or TF-IDF. However, we decided to use texts_to_sequences because we wanted to highlight, later in this section, how Keras Word Embedding works with GloVe. The following code shows how texts_to_sequences works:

words = "congratulations brand new bill gates lottery winner send account number transfer money you"
word_encoded = tokenizer.texts_to_sequences(words)
print(word_encoded)

This produces one integer for each word:

[85, 105, 177, 2521, 19, 2118, 317, 678, 85, 654, 812, 645, 145], dtype=int32

Padding

By padding, we determine the maximum length *maxlen* the string entered into the DL algorithm must have. If the words on the website are longer than *maxlen*, this string will be truncated until the length is *maxlen*. On the other hand, if the length of the website is less than *maxlen*, then the spaces in which there are no words will be filled by a PAD tag until the length *maxlen* is obtained. Let us analyze the following code:

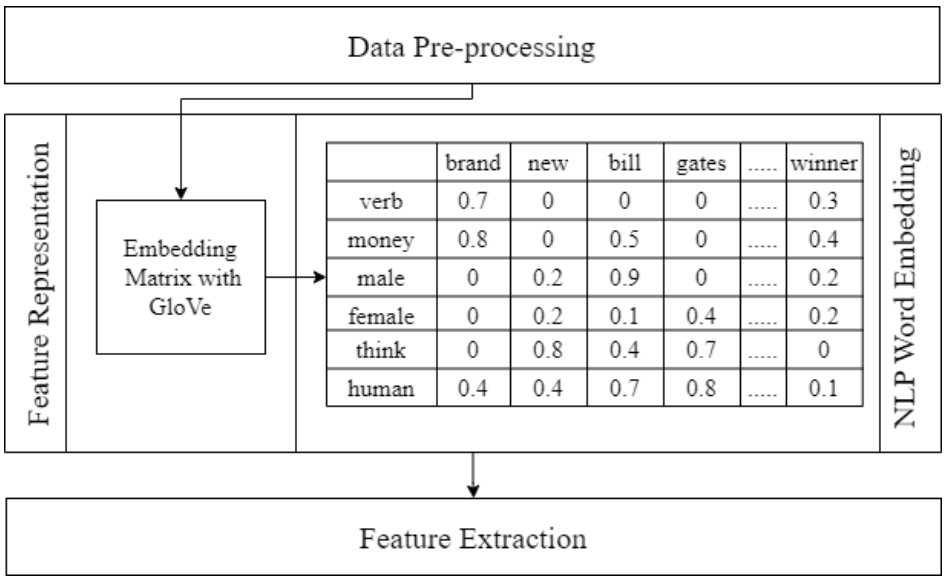
maxlen = 200
tokens = [85, 105, 177, 2521, 19, 2118, 317, 678, 85, 654, 812, 645, 145]
word_sequences = pad_sequences(tokens, padding = 'post', maxlen = maxlen)
print(word_sequences)

It will produce the following sequence until all spaces to the right of the sequence are filled with zeros. If the sequence is more than 200, then the sequence will be truncated to 200:

word_sequences = [85, 105, 177, 2521, 19, 2118, 317, 678, 85, 654, 812, 645, 145, 0, 0, 0, 0, 0, ... , 0]

5.1.3. Feature Representation with Keras Embedding and GloVe

Keras Embedding Layer with pre-trained GloVe word embeddings works by mapping each word in the input stream to a pre-trained vector representation, which is learned based on the distributional properties of words in a large text corpus. The main idea of our model is to obtain the semantic and syntactic importance of the text of the web pages before the DL algorithms parse it. Hence, we used Keras Embedding with the pre-trained GloVe word embeddings dataset [37]. Figure 5.4 shows an example of how, from a list of words or their coded values, with the use of Keras Embedding and GloVe, a matrix is obtained in which values are stored that indicate the relationship that would exist between a word (of those used in our example) of a row with that of a column, considering that a value close to zero indicates that there is practically no relationship. In contrast, a value close to one indicates a high relationship. For example, if in the matrix of Figure 5.4, we analyze the word *gates*, we see that it has a high probability that it is related to a *thing* (0.7) or that it is related to a *human* being (0.8), but it has a very low probability that it is a verb (0).



	brand	new	bill	gates	.....	winner
verb	0.7	0	0	0	.....	0.3
money	0.8	0	0.5	0	.....	0.4
male	0	0.2	0.9	0	.....	0.2
female	0	0.2	0.1	0.4	.....	0.2
think	0	0.8	0.4	0.7	.....	0
human	0.4	0.4	0.7	0.8	.....	0.1

Figure 5.4: Feature representation

When using pre-trained GloVe word embeddings with the Keras Embedding Layer, the pre-trained embeddings are loaded into memory as a dictionary in which each word is associated with a pre-trained vector representation. An embedding matrix is then created for the vocabulary by looking up the pre-trained vectors for each word. This embedding matrix is used to initialize the weights of the Keras Embedding Layer. In our study, we created the following Keras Embedding Layer with GloVe:

embedding_layer = Embedding(vocab_size, output_dim = 100, weights = [embedding_matrix], trainable = False)(deep_inputs)

where:

vocab_size = 144,236.

embedding_dim = 100.

weights = embedding_matrix obtained with the dataset glove.6B.100d.txt.

trainable = False, because it was already initialized with glove.6B.100d.txt.

(deep_inputs). This means that the input will be of type 2D and is expected to be a matrix of integers where each row represents a sequence of tokens.

5.1.4. DL Algorithms Execution

The final step to test our model's accuracy and mean accuracy is to input the resulting Keras embedding data into each DL algorithm, with which our model is trained and tested. As additional work in our research, we decided to use four DL algorithms: LSTM, BiLSTM, GRU, and BiGRU, to determine which algorithm best fits our model, for which we developed the respective code for each one in Python. Each algorithm setup is presented in the following sections.

The four DL algorithms that we have decided to use are variants of RNN, considering that these algorithms have a memory, which makes them particularly suitable for processing data series with a temporal order or structure, such as words in a text. According to [40], the two outstanding RNN algorithms for dealing with time series are LSTM with 100% and GRU with 99.99%, respectively, in evaluating the Receiver Operating Characteristic (ROC) curve on phishing URLs. That is why we decided to study these algorithms, LSTM and GRU, but on text. We also checked their respective bi-directional approaches, BiLSTM and BiGRU. Thus, to evaluate our model and determine which of the four algorithms is the best for detecting phishing attacks, based on the content of the websites, we will use LSTM, BiLSTM, GRU, and BiGRU.

The main idea of using these algorithms is to take advantage of the intrinsic richness in the composition of sentences and the text itself. We consider their semantic and syntactic meaning, i.e., not to use only local feature representations but to go beyond that, using non-spatial features, such as time series. In addition, we mainly use BiLSTM and BiGRU because these algorithms reward and adjust in both forward and backward directions. This way, the analyzed text's semantic and syntactic meaning can be obtained more accurately.

Also, because our issue is a binary classification problem, we use binary_cross_entropy as an optimizer. Also, since our dataset was unbalanced, we used K-fold cross-validation.

5.2. Experimental Setup

Once the model has pre-processed the dataset, the DL algorithm will remain to run on the data obtained. However, our work also aimed to determine which DL algorithm best fits the data obtained from our model. For this reason, we decided to experiment with four different DL algorithms: LSTM, BiLSTM, GRU, and BiGRU. This section details the hardware, software, and dataset used and the configuration of the four selected DL algorithms.

5.2.1. Hardware and Software Environment

To execute our experiment, we used a RIG server with Python 3.5.2 on Jupyter Notebook 6.0.2 and the libraries Keras, NLTK, NumPy, pandas, request, sci-kit learn, and TensorFlow. Table 5.1 presents the RIG features of each component of the hardware environment.

Table 5.1: RIG features.

Component	**Model**
Processor	AMD Ryzen Threadrippe 2920X
RAM	16 GB Crucial Ballistix DDR4-3000
Video card	16 GB Phantom Gaming X Radeon VII
SSD	500 GB Crucial SSD M.2 NVMe
HD	3 TB Western Digital HDD Purple
Mainboard	ASUS ROG Zenith Extreme Alpha

5.2.2. Dataset

We used the freely available dataset Phishload [115], which was then decompressed into a SQL-like file, opened in HeidySQL, and exported to a CSV format, which is more Python-friendly.

This dataset is composed of three tables, of which we used the website table, from which we extracted two columns:

- htmlContent column, which contains the HTML code of all the web pages.

- isPhish column, which indicates whether a website is phishing or ham.

This dataset comprises 10,488 rows, but after deleting the rows containing null fields, the dataset was reduced to 10,373 in total, of which 9198 phishing rows and 1176 ham rows were obtained in the end. While the dataset is unbalanced, we used performance measurements for unbalanced data in the evaluation process [118]. Hence, during the execution and to demonstrate the experiment's validity, we used the K-fold cross-validation technique with K=5, considering that the dataset was divided into 80% for training and 20% for testing.

5.2.3. DL Algorithms Setup

According to [3], RNN and CNN are the most used DL algorithms for phishing attack detection. Still, RNN has a different characteristic from CNN since it is developed for time-series data [119]. RNN has directional connections that allow it to compute the next step, building on previous steps [120]. Thus, RNN is widely used in NLP and fits our study very well because our analysis is oriented to the sequence of words obtained from web pages. A disadvantage of a simple RNN is that it cannot easily find meaningful connections that go beyond 10-time steps [99]. There are RNN-derived algorithms that can solve this problem. Therefore, we have decided to use the following four RNN algorithms for our study: LSTM, BiLSTM, GRU, and BiGRU.

LSTM

This is a variation of the RNN [121]. Unlike a simple RNN algorithm, which can perform computations based on a few word sequences, LSTM can calculate based on recent and non-recent words. In other words, it can determine the importance of the data with more than 1000 time steps between them [99]. Figure 5.5 shows the LSTM algorithm setup.

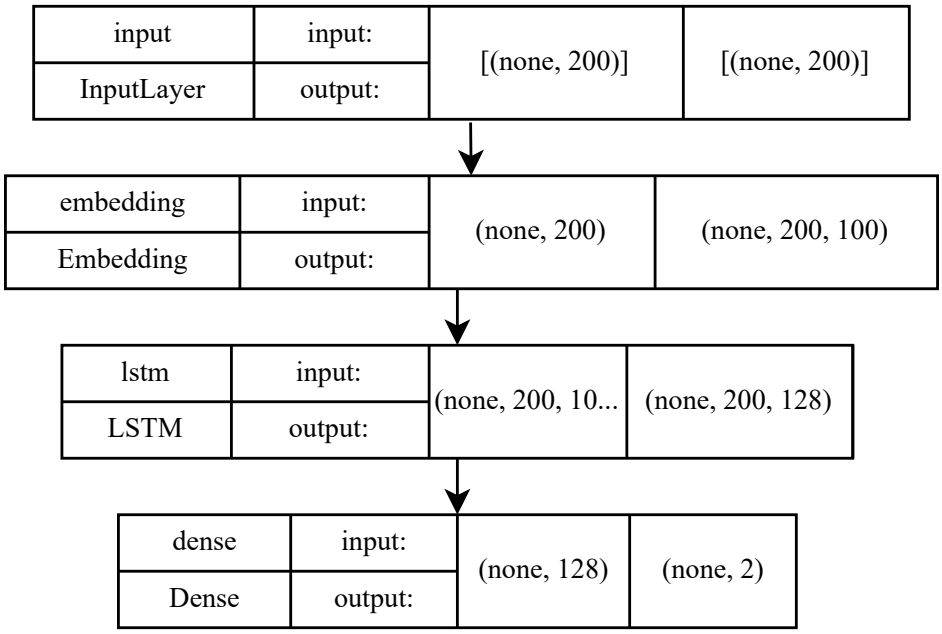


Figure 5.5: LSTM algorithm setup

BiLSTM

This is the RNN layer, a sequence-processing model with two LSTMs: a forward LSTM and a reverse LSTM. By this back-and-forth process, BiLSTM increases the amount of information available to the network, further improving the context of a study such as ours, where it is essential to know which word precedes and follows another word [122]. Figure 5.6 shows the BiLSTM algorithm setup.

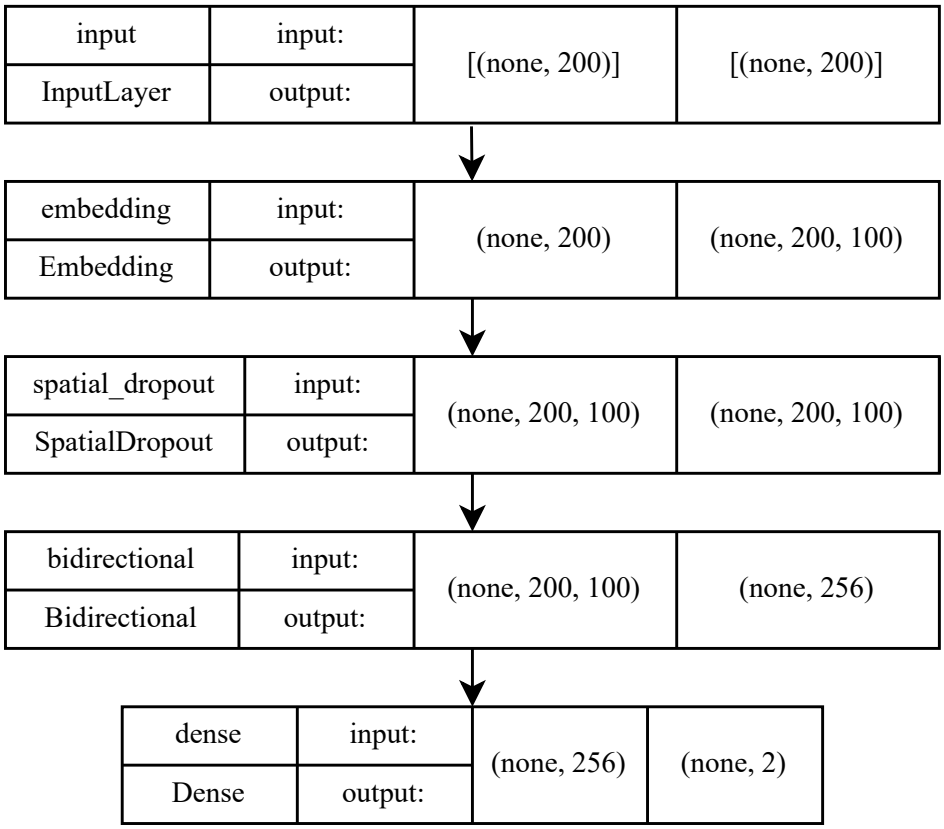


Figure 5.6: BiLSTM algorithm setup

GRU

This type of RNN is like an LSTM network but with a forget gate. It also has fewer parameters than LSTM because it does not have an output gate. The performance of GRU networks is similar to that of LSTM when it comes to NLP [123]. Figure 5.7 shows the GRU algorithm setup.

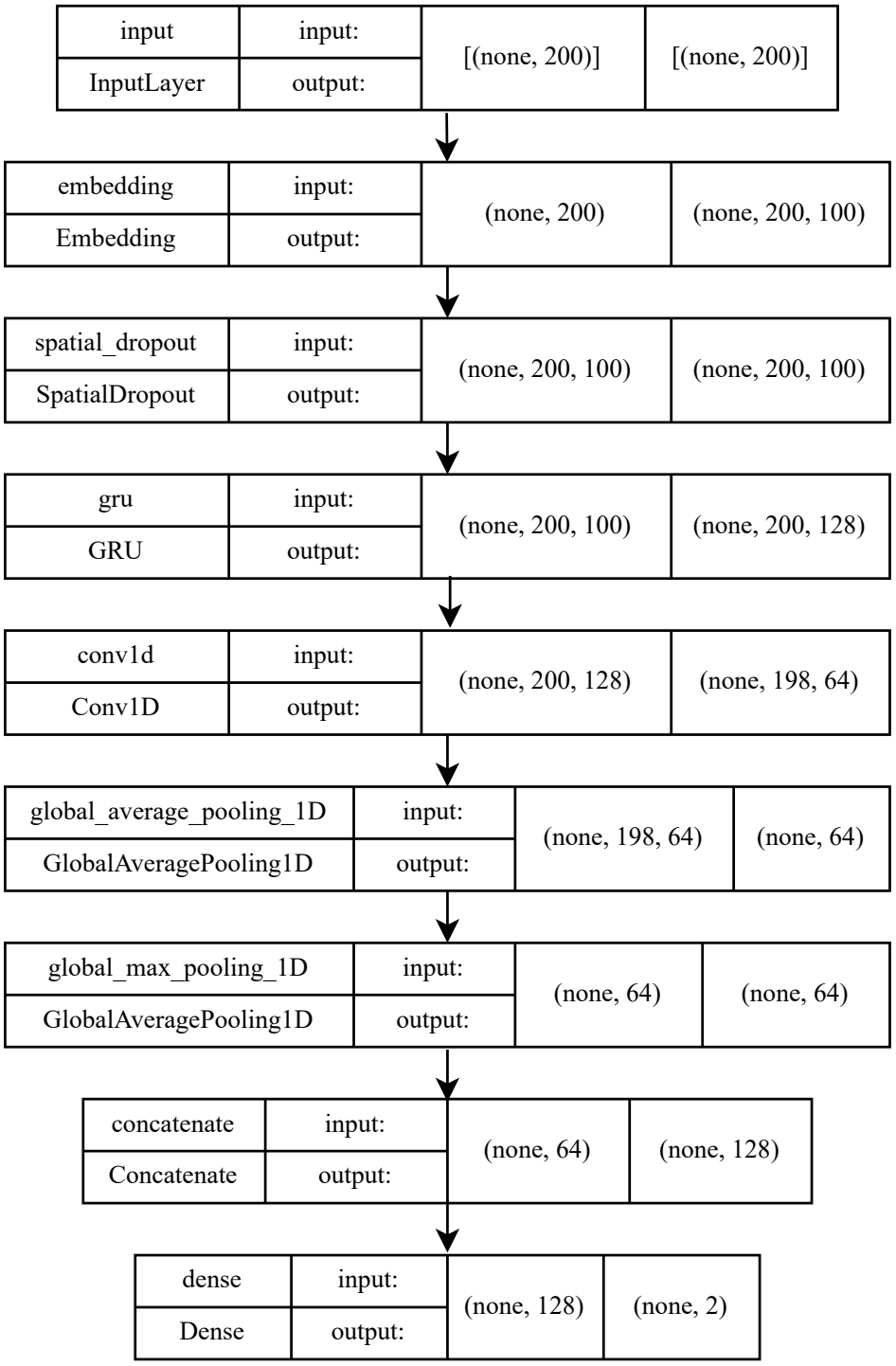


Figure 5.7: GRU algorithm setup

BiGRU

A GRU network employs recurrence to store and retrieve information for long periods, but in practice, its performance could be better because the network only accesses past information [124]. Thus, to solve this information access problem, BiGRU has a future layer in which

the data sequence is in the opposite direction. Therefore, this network uses two hidden layers to extract past and future information. These hidden layers are then connected into a single output layer [125]. These characteristics enable the bidirectional structure to assist the RNN in extracting more information and, consequently, improve the performance of the learning process. Figure 5.8 shows the BiGRU algorithm setup.

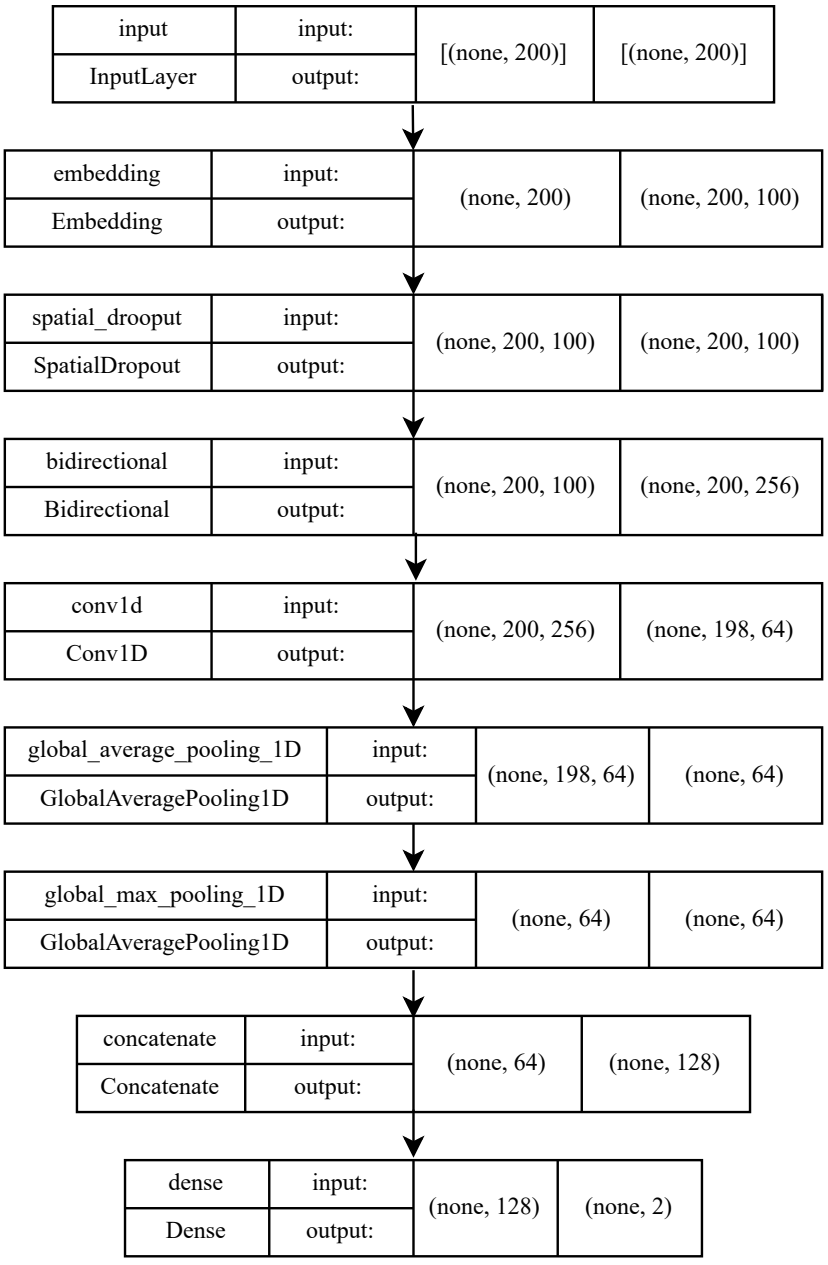


Figure 5.8: BiGRU algorithm setup

5.2.4. Performance Metrics

We adopted commonly used quality metrics to compare the correctness of the classification of the detection model. We used the following terms for determining the quality of the classification models [126]:

- *True Positive (TP)*: the number of data items correctly classified to the positive class.
- *True Negative (TN)*: the number of data items correctly classified to the negative class.
- *False Positive (FP)*: the number of data items wrongly classified to the positive class.
- *False Negative (FN)*: the number of data items wrongly classified to the negative class.

When dealing with unbalanced data, choosing appropriate evaluation metrics that consider the class distribution in the dataset is important. We used the following metrics:

- Precision: Precision measures the proportion of true positives among the predicted positives. This metric helps minimize false positives or when the positive class is rare.

$$Precision = TP/(TP + FP) \tag{1}$$

- Recall: Recall measures the proportion of true positives among the total number of actual positives. This metric is useful when minimizing false negatives or when the positive class is important.

$$Recall = TP/(TP + FN) \tag{2}$$

- F1-score: the F1-score is the harmonic mean of the precision and recall and is a good metric to use when there is an uneven distribution of classes in the dataset.

$$F1\text{-}Score = 2 \times ((Precision \times Recall)/(Precision + Recall)) \tag{3}$$

- Micro avg: calculates the overall average precision score over all classes, weighting each instance by its weight.
- Weighted avg: calculates the score by averaging the results for each class, weighting each result by the number of instances of that class in the dataset.

- K-fold cross-validation: Since the data are not balanced, we used the K-fold cross-validation technique to calculate the mean accuracy. This technique is also helpful in avoiding overfitting the training data. This technique is a suitable metric for determining which of the four DL algorithms performs best.

5.3. Chapter Summary

This chapter determined the methods, materials, and tools to develop a model to detect Phishing attacks. This model must use Deep Learning and natural language processing on text obtained from web pages. Thus, in the first section of the chapter, each step followed in the data preprocessing was determined, including using the GloVe dictionary. It is agreed to run the experiment's LSTM, BiLSTM, GRU, and BiGRU algorithms. This chapter's second section determined the initial hyper-parameters configuration to run the experiments with the four selected algorithms: LSTM, BiLSTM, GRU, and BiGRU. Next, a default configuration was established that will be refined in Chapter 8. The metrics used are Precision, Recall, F1-score, micro average, and Weighted average. The K-fold cross-validation was used because the data were not balanced. In the next Chapter, the experiment is carried out.

Chapter 6

Results of the Implementation of the Phishing Detection Model

Contents

6.1. Results

6.1.1. Results of Test loss, test AUC, training loss, and training AUC.

Table 6.1 shows the test loss, test AUC, training loss, and training AUC obtained with each DL algorithm and each L. It was observed that GRU and BiGRU always gave better test accuracy than the other algorithms. BiLSTM showed values close to GRU or BiGRU; however, LSTM gave the worst results.

Table 6.1: Test loss, test AUC, training loss, and training AUC with L = 1000, 500, and 200.

Algorithm	L	Test Loss	Test AUC	Train Loss	Train AUC
LSTM	1000	0.27	0.92	0.27	0.93
BiLSTM	1000	0.17	0.98	0.12	0.99
GRU	1000	0.14	0.99	0.09	0.99
BiGRU	1000	0.15	0.96	0.07	1
LSTM	500	0.23	0.95	0.23	0.95
BiLSTM	500	0.17	0.98	0.12	0.99
GRU	500	0.14	0.99	0.09	0.96
BiGRU	500	0.14	0.99	0.07	1
LSTM	200	0.19	0.97	0.18	0.97
BiLSTM	200	0.19	0.98	0.10	0.99
GRU	200	0.16	0.98	0.09	0.99
BiGRU	200	0.17	0.98	0.07	1

6.1.2. Area Under the ROC Curve (AUC).

Although the AUC is not as appropriate to define the accuracy of an algorithm, it is important to compare the performance of the four DL algorithms. Table 6.2 shows the AUC calculated with each length L. Our model is of binary type since only two classes are evaluated (if the AUC is close to 0, indicating ham; or if the AUC is close to 1, indicating phishing). It was observed that GRU and BiGRU gave better results than LSTM and BiLSTM. Even at L = 200,

GRU scored better. In addition, we also controlled the execution time of each DL algorithm in our model and observed that GRU was the one that was trained the fastest because it was trained in 240 seconds.

Table 6.2: Area Under the ROC Curve (AUC) with L = 1000, 500, and 200.

Algorithm	**L**	**AUC = 0 (Ham)**	**AUC = 1 (Phishing)**	**Time (s)**
LSTM	1000	0.74	0.72	1480
BiLSTM	1000	0.94	0.94	1660
GRU	1000	0.96	0.96	1320
BiGRU	1000	0.96	0.96	1860
LSTM	500	0.81	0.80	700
BiLSTM	500	0.94	0.94	800
GRU	500	0.96	0.96	600
BiGRU	500	0.96	0.96	840
LSTM	200	0.91	0.91	300
BiLSTM	200	0.94	0.94	320
GRU	200	0.96	0.96	240
BiGRU	200	0.95	0.95	320

6.1.3. Micro avg, weighted avg, and F1-score.

Table 6.3 shows the computed micro average and weighted average metrics. In the three runs of the experiment with L = 1000, 500, and 200, it was again observed that GRU and BiGRU prevailed over LSTM and BiLSTM. The weighted average is the most appropriate metric for unbalanced binary data. Still, even for those, GRU was considered equal to Bi-GRU. To continue our experiment with F1-Score, we have used L = 200 only because it represents 58.02% of our dataset. We do not use values below 200 words because the metrics decrease considerably.

Table 6.3: Micro avg, weighted avg, and F1-score.

Algorithm	L	Micro Avg	Weighted Avg	F1-Score
LSTM	1000	0.91	0.88	
BiLSTM	1000	0.98	0.96	
GRU	1000	0.99	0.98	
BiGRU	1000	0.99	0.98	
LSTM	500	0.94	0.92	
BiLSTM	500	0.98	0.97	
GRU	500	0.99	0.98	
BiGRU	500	0.99	0.98	
LSTM	200	0.97	0.95	0.93
BiLSTM	200	0.98	0.96	0.74
GRU	200	0.98	0.97	0.94
BiGRU	200	0.98	0.97	0.94

6.1.4. Mean accuracy in percent with K-fold cross-validation with K = 1 to K = 5.

To define which DL algorithms worked best with the data embedded with GloVe, we executed the four algorithms using the K-fold cross-validation technique, where K = 5 and shuffle = true. Hence, the results obtained in each K-fold and the mean accuracy obtained in each algorithm can be seen in Table 6.4.

Table 6.4: Mean accuracy in percent with K-fold cross-validation with K = 1 to K = 5.

Algorithm	K = 1	K = 2	K = 3	K = 4	K = 5	Mean
LSTM	94.12	95.76	96.48	98.36	98.84	96.71
BiLSTM	94.26	95.95	97.96	98.75	99.08	97.20
GRU	95.03	95.90	98.26	98.41	99.28	97.29
BiGRU	95.22	96.53	98.55	98.84	98.84	97.39

6.2. Discussion

Based on what was stated in the previous section, we present the graphical analysis performed when executing each algorithm with an inserted Length (L) of 200 words. As seen in Figures 6.1 to 6.4, the four models worked well because they were generalizing correctly. Consequently, each model's training accuracies' values were close to their respective validation accuracies' values.

6.2.1. NLP-LSTM

As can be seen in Figure 6.1, the test accuracy (97%) and training accuracy (97%) obtained with LSTM were acceptable. Again, it can be seen that the validation accuracy line advanced very close to the training accuracy; However, at Epoch 10, there was a drop in accuracy to less than 95%. This drop may be due to an overfitting problem. Among the four DL algorithms, LSTM was the worst performer.

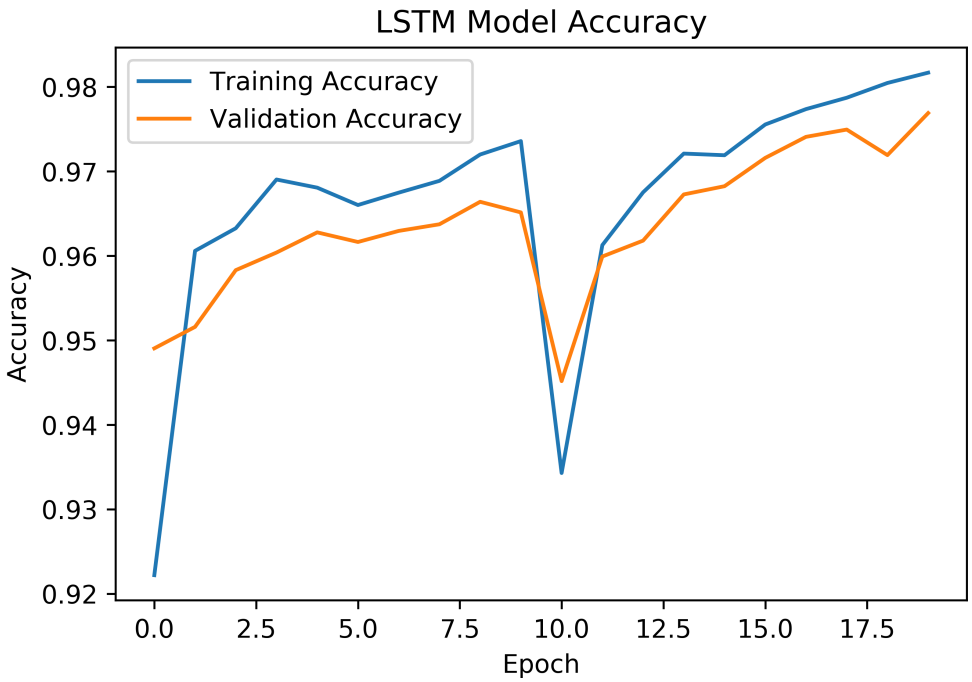


Figure 6.1: LSTM accuracy

6.2.2. NLP-BiLSTM

Figure 6.2 shows that the training accuracy (98%) and validation accuracy (99%) obtained with BiLSTM outperformed those obtained with LSTM. Although there was also a decay at Epoch 10, this decayed slightly (to 97%). At the end of the run, at Epoch 17, there was a decrease in the validation accuracy, indicating possible overfitting.

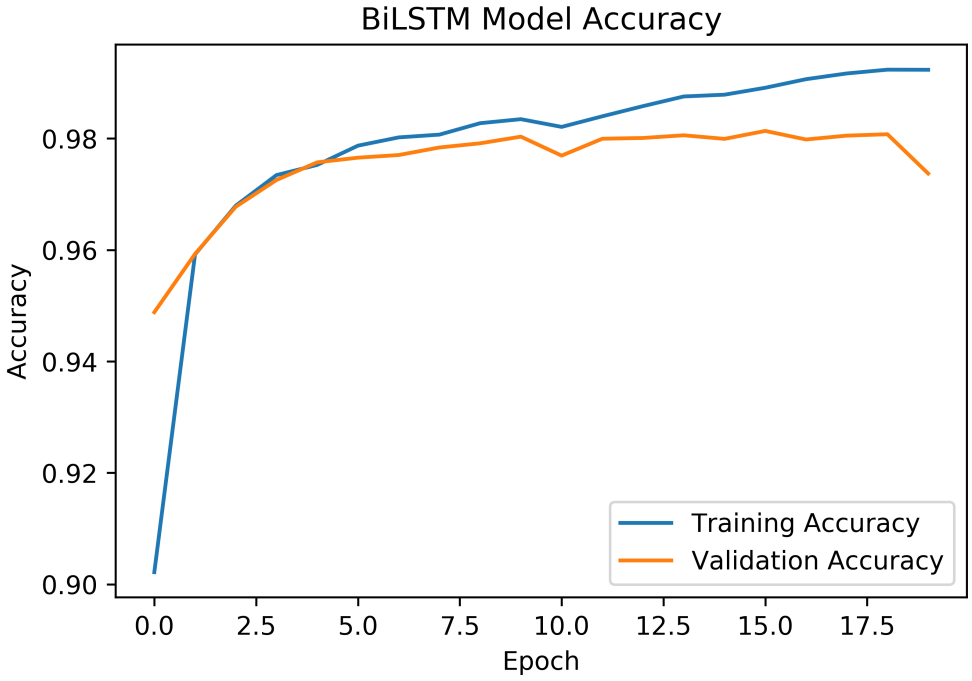


Figure 6.2: BiLSTM accuracy

6.2.3. NLP-GRU

Figure 6.3 shows that the training accuracy (98%) and validation accuracy (99%) improve considerably. Negative slopes are not observed in the two lines but are constant growth. In addition, although the validation accuracy declined a little at the end of the processing, compared to the validation accuracy, this decline was minimal, remaining close to the training accuracy line. Additionally, the processing time was 240 s, thus surpassing the processing time of LSTM with 300 s and BiLSTM with 320 s.

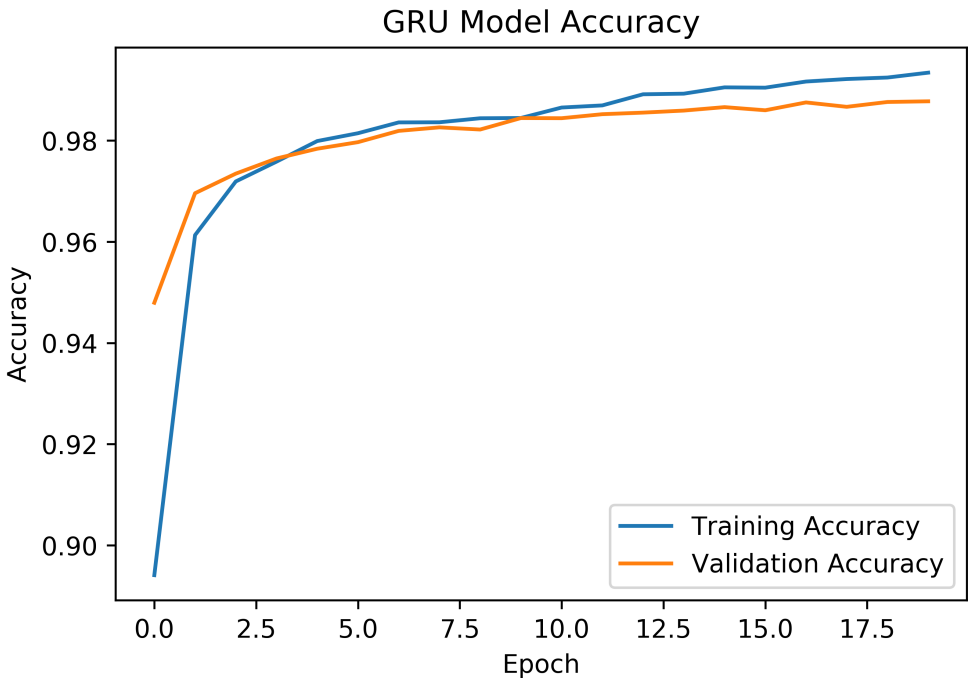


Figure 6.3: GRU accuracy

6.2.4. NLP-BiGRU

Figure 6.4 shows that the NLP-BiGRU combination was the one that gave the best results for the training accuracy (100%). However, the validation accuracy obtained (98%) did not vary concerning BiLSTM (98%) and GRU (98%). On the other hand, the time to achieve its training execution was 320s, i.e., 33% more than the time GRU processed.

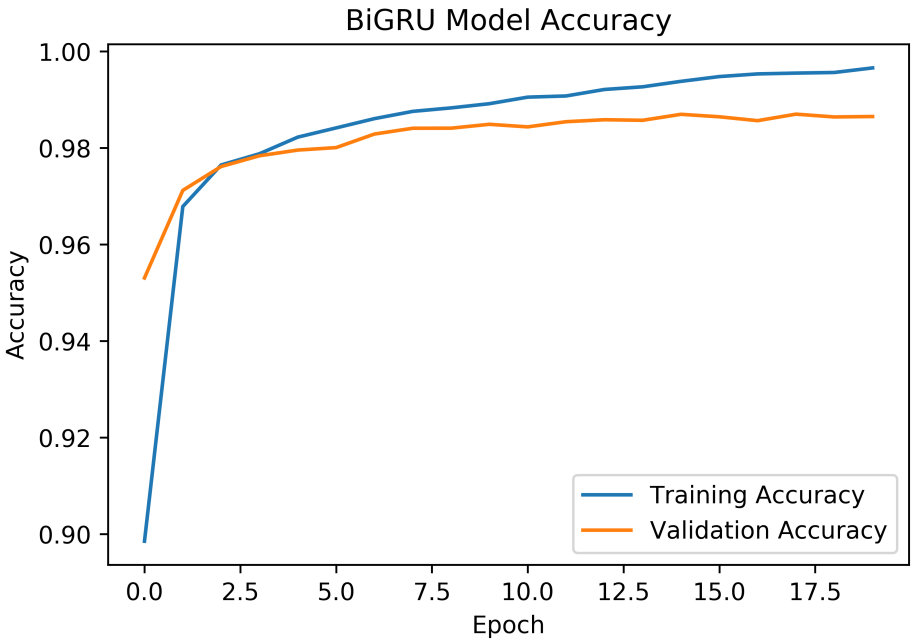


Figure 6.4: BiGRU accuracy

6.2.5. Graphical Comparison of the Four Models

Figure 6.5 compares the accuracy obtained for each epoch in the four models. It can be seen that LSTM behaved the worst, even showing erratic behavior, with negative slopes in several periods, highlighting a very pronounced negative slope at Epoch 10, with an accuracy below 95%. The BiLSTM model gave better results than LSTM. Although the training accuracy improved notably, according to Figure 6.2, the same did not occur with the validation accuracy, which had a negative slope at Epoch 10 and a pronounced negative slope at the end of Epoch 20. GRU and BiGRU gave better results in training than those offered by LSTM and BiLSTM since it was observed that, in both of them, an accuracy that constantly increased from the first epoch was obtained. Analyzing GRU versus BiGRU, it was observed that BiGRU gave better results, obtaining in the last Epoch 20 100% for its training accuracy, over the 99% obtained in GRU. Even in Figure 6.6, it can be seen that the model with the lowest loss was BiGRU, followed by GRU. On the other hand, of the four algorithms analyzed, the one that performed worst was LSTM.

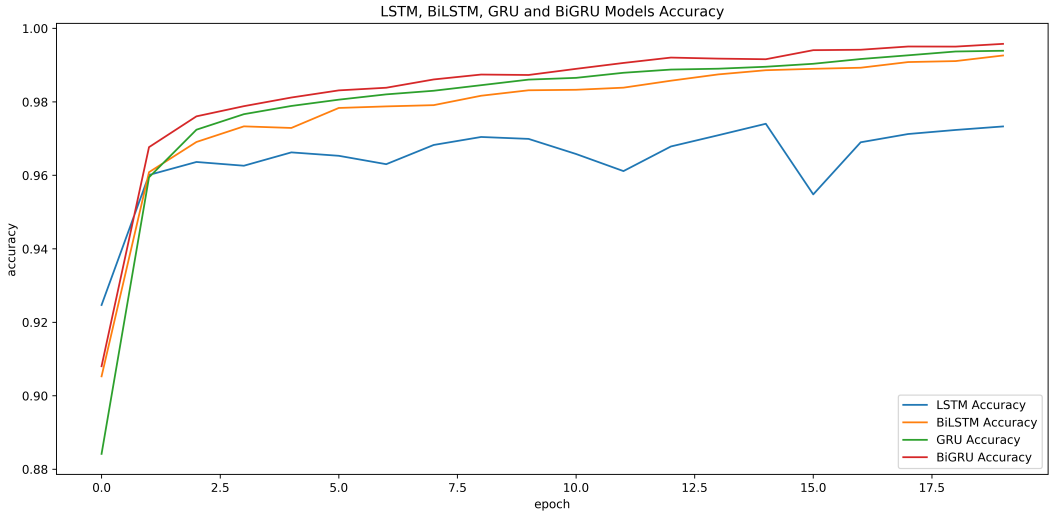


Figure 6.5: Accuracy comparison of the four algorithms

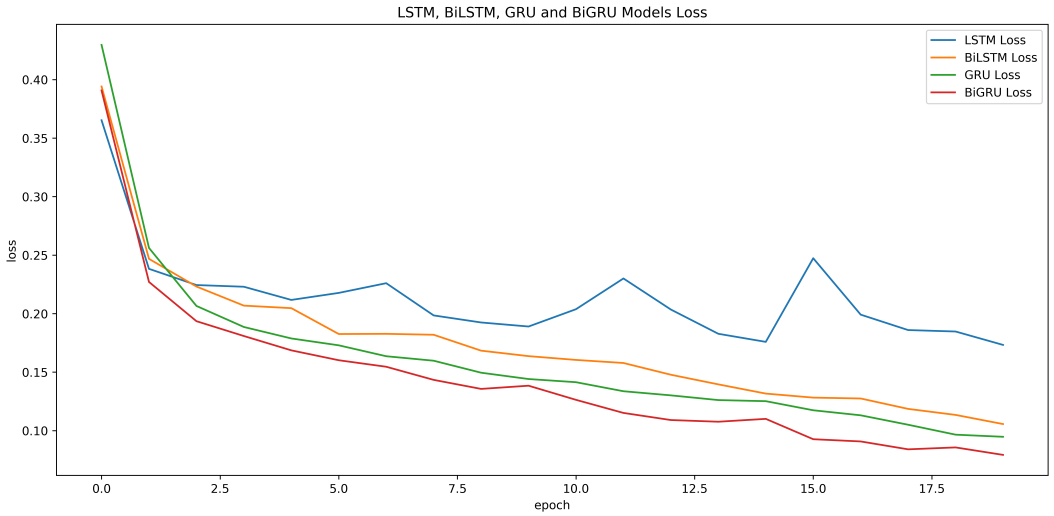


Figure 6.6: Loss comparison of the four algorithms

To define which DL algorithms worked best with the data embedded with GloVe, we executed the four algorithms using the K-fold cross-validation technique, where K = 5 and shuffle = true. The execution results in Table 6.4 indicate that our model is acceptable, as all four algorithms performed with a mean accuracy above 96.70%. The DL algorithm with the best results was the BiGRU algorithm, with 97.39%.

6.3. Chapter Summary

Based on the materials and methods of the previous chapter, experiments were carried out in this chapter to obtain a model to detect a phishing attack with Deep Learning and

natural language processing with GloVe. Thus, in the first section of this chapter, the results of the metrics obtained in the execution of the model with the LSTM, BiLSTM, GRU, and BiGRU algorithms are shown, with 1,000, 500, and 200 words. In the second section, the results obtained in the execution of the models with the four Deep Learning algorithms are discussed.

This experiment determined that the algorithm that produced the best results was BiGRU, with up to 100% accuracy obtained in epoch 20. However, due to the unbalanced data, the algorithms were evaluated with k-fold cross-validation with k=5, which resulted in all four algorithms reaching a mean accuracy of 96.70%, with BIGRU reaching an accuracy of 97.39%. Based on the results obtained, in the following chapter, BiGRU will be used as the algorithm for which the tuning will be carried out and on which an extension will be made to install in the Chrome browser.

Chapter 7

Fine-Tuning and Implementation of the NDLP Model

Contents

The previous chapter proposed a model to detect phishing attacks through deep learning and NLP. In addition, it was determined that the Deep Learning algorithm that provides the most remarkable accuracy for detecting this type of attack on the text of web pages is

BiGRU; however, a fine-tuning of the hyper-parameters of the BiGRU algorithm and the NLP parameters still needs to be performed. Thus, this section's first objective is to determine the optimal hyper-parameters to detect phishing attacks with high accuracy and little computational expense. Once the model has been optimized by adjusting hyper-parameters, the second objective of this chapter is to develop an extension that can be installed in the Chrome browser. This extension allows the end user to determine in a friendly way whether a page is Phishing. Hence, this section is composed of two parts. The first part shows the results obtained in the model tuning and determines the optimal hyper-parameters for this tuning. The second part develops the steps to make the extension based on the tuned algorithm and shows how it works.

7.1. Hyper-Parameters to Fine-Tune the NLP and DL Phishing Attack Detection Application

The entire tuning experiment was performed in Jupyter Notebook and can be reproduced with the code shared in the following GitHub link:

https://github.com/debenavides/NDLP-hyper-parameter-tuning/

In the comments of the shared code above, each hyper-parameter and the values used are shown. The tuning was performed on the model proposed in the study [2] because it meets the assumptions of predicting a phishing attack using NLP and DL on the text obtained from the body of the web pages. Before starting, it should be mentioned that in the study [2] several aspects were analyzed, detailed as follows, and will be the basis of our experiment. It was determined in [2] that the algorithm that gave the best results for this type of problem was BiGRU, surpassing the mean accuracy of LSTM, BiLSTM, and GRU. This research offers a high-accuracy, lightweight model that can be installed on an ordinary computer or mobile device. Our tuning aims to offer a lightweight application without sacrificing accuracy. The steps to perform the tuning of the hyper-parameters are shown in Figure 7.1

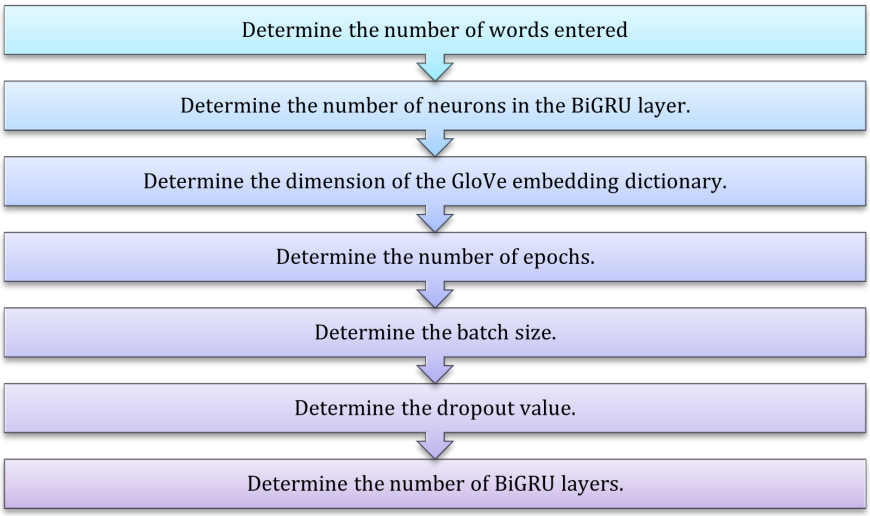


Figure 7.1: Steps to determine the optimal hyper-parameters of NDLP Phishing.

There are three main strategies to determine the hyper-parameters of machine learning algorithms: grid search, random search, and manual search. For our study, we followed the manual search method [127]. Although Grid Search and Random Search could obtain higher hyper-parameter resolution, the computational cost is very high. Furthermore, according to [128], most hyper-parameters that are analyzed will not cause the accuracy of the analyzed algorithm to increase significantly. Therefore, we have adopted the manual hyper-parameter search approach based on expert knowledge, intuition, and experience. Table 7.1 shows the hyper-parameters with each value evaluated.

Table 7.1: hyper-parameters to evaluate.

hyper-parameter	**Values**			
Number of words	100	200	500	1000
BiGRU neurons	32	64	128	256
Dimension of GloVe	-	50	100	200
Number of epochs	-	5	10	20
Batch size	32	64	128	256
Dropout value	-	0.1	0.5	0.7
BiGRU layers	-	-	1	2

7.1.1. Number of words to enter the algorithm

Analyzing our dataset, as shown in Figure 7.2, three red lines are drawn, corresponding to L = 200, L = 500, and L = 1000 words per website, in which 9.78% of 10,373 websites had more than 1000 words. For this reason, only a max length L of 1000 words and below was taken for the NLP analysis. Thus, three values of L were defined to analyze which value works best: L = 1000 represents 90.22% of the data; L = 500, which is 81.33% of the data; L = 200, which means 58.02% of the data.

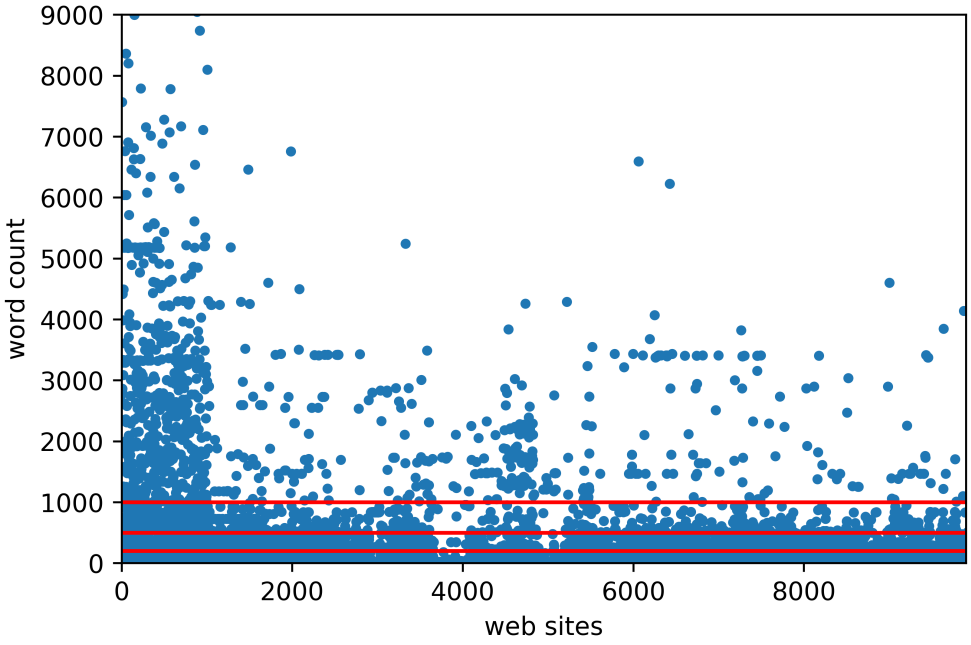


Figure 7.2: Word length distribution over the entire dataset

Due to the different text lengths L obtained from each web page analyzed, the four algorithms were run with three different values of L (200, 500, and 1000) to test which of the lengths L of words performed better. The testing results of the algorithms were presented in the article [2] with four text inputs of 100, 200, 500, and 1000 words, and it was determined that the optimal number of words to be entered was 200.

7.1.2. Number of Neurons in the BiGRU Layer

Based on the experiment [2], in which 128 neurons were used in the BiGRU layer, we evaluated the mean accuracy obtained with 32, 64, 128, and 256 neurons in the current study. The results obtained can be seen in Table 7.2.

Table 7.2 shows that the K-fold mean accuracy obtained with 32 neurons significantly differs from that obtained with 64, 128, and 256 neurons. This difference is because the mean accuracy curve tends to stabilize while the neurons increase. On the other hand, with 256 neurons, the mean accuracy rises slightly but with a high computational cost, which means that 32 and 256 neurons are discarded; henceforth, experiments are performed only with 64 and 128 neurons.

Table 7.2: BiGRU neurons.

BiGRU neurons	**K-fold mean accuracy**
32	0.9680
64	0.9713
128	0.9737
256	0.9757

7.1.3. Dimension of the GloVe Embedding Dictionary

The following tuning step was determining the best dimension of the GloVe dictionary [37], considering that we have GloVe dimensions of 50, 100, 200, and 300. The results of the tests with vectors of 50, 100, and 200 dimensions are shown in Table 7.3.

Table 7.3: GloVe dimensions.

BiGRU neurons	**GloVe Dimension**	**K-fold mean accuracy**
64	50	0.9685
64	100	0.9703
64	200	0.9749
128	50	0.9681
128	100	0.9736
128	200	0.9758

As shown in Table 7.3, the algorithms' performance was evaluated with 50, 100, and 200 dimensions of the GloVe dictionary (300 was unnecessary), with 64 and 128 BiGRU neurons. The value obtained with 100 and 200 GloVe dimensions does not vary significantly from that obtained with a 50-dimensional vector. This may be because the model is relatively

simple. Since a lightweight model is required as a premise, a size of 50 GloVe dimensions is adopted as the optimal value.

7.1.4. Number of Epochs

So far, ten epochs have been used for testing. The model is evaluated with 5, 10, and 20 epochs to improve the mean accuracy. The results of the model execution with the different epochs are shown in Table 7.4.

Table 7.4: Epochs.

Epochs	**BiGRU neurons**	**GloVe Dimension**	**Elapsed time (min)**	**K-fold mean accuracy**
5	64	50	0:12	0.9905
10	64	50	0:23	0.9896
20	64	50	0:46	0.9844

Table 7.4 shows that the accuracy decreases as the epochs increase, possibly due to overfitting. Even with five epochs, a mean accuracy of 0.9905 is obtained. From there, five epochs are adopted for the model.

7.1.5. Batch Size

A larger batch size can lead to greater generalization; however, this requires more memory capacity and longer training time, so different batch size values of 32, 64, 128, and 256 are analyzed. See the results in Table 7.5.

Table 7.5: Batch size.

Batch size	**Epochs**	**BiGRU neurons**	**GloVe Dimension**	**Elapsed time (min)**	**K-fold mean accuracy**
32	5	64	50	0:51	0.9851
64	5	64	50	0:26	0.9896
128	5	64	50	0:13	0.9844
256	5	64	50	0:09	0.9611

Table 7.5 shows that the mean accuracy value obtained for batch size = 256 is considerably lower than the other values. In addition, the value obtained with batch size = 64 offers better values than the other batch size values.

7.1.6. Dropout Value

It is essential to obtain an adequate dropout size to reduce overfitting without affecting performance and speed [129]; for this, an evaluation is performed at low, medium, and high dropout values. The values obtained with dropouts 0.1, 0.5, and 0.7 can be seen in Table 7.6.

Table 7.6: Dropout size.

Dropout	**Batch size**	**Epochs**	**BiGRU neurons**	**GloVe Dimension**	**Elapsed time (min)**	**K-fold mean accuracy**
0.1	64	5	64	50	0:12	0.9635
0.5	64	5	64	50	0:12	0.9454
0.7	64	5	64	50	0:11	0.9384

Table 7.6 shows that the elapsed time in the execution with the three dropout values analyzed practically remains unchanged; however, the mean accuracy obtained with the 0.1 dropouts is better than that obtained with the other two dropout values.

7.1.7. Number of BiGRU Layers

After finding the optimal hyper-parameters for one BiGRU layer, we added a layer to test for improved accuracy. The results for one and two layers of BiGRU are presented in Table 10.

Table 7.7: BiGRU layers.

BiGRU layers	**Dropout**	**Batch size**	**Epochs**	**BiGRU neurons**	**GloVe dimension**	**Elapsed time (min)**	**K-fold mean Accuracy**
1	0.1	64	5	64	50	0:11	0.9635
2	0.1	64	5	64	50	0:19	0.9454

Based on the results shown in Table 7.7, we found that increasing the number of layers in

our DL model did not improve the mean accuracy. Furthermore, the training time increased by seven minutes compared to the previous model. We concluded that adding more layers to the model is unnecessary and may lead to increased complexity, longer processing times, and decreased accuracy. This decrease may be because the data handled by our model is not necessarily very complex.

7.1.8. Tuned Hyper-Parameters

After tuning the model, Table 7.8 shows the values of the hyper-parameters obtained in our proposal.

Table 7.8: Tuned hyper-parameters for the NDLP application.

	BiGRU layers	**Dropout**	**Batch size**	**Epochs**	**BiGRU neurons**	**GloVe Dimension**
Our tuned model NDLP	1	0.1	64	5	64	50

7.2. Application Based on the Fine-Tuned Attack Detection Model using NLP and DL

In this section, we present the Google extension developed to implement the fine-tuned model.

7.2.1. Coding and Testing

The code files used to develop the Google extension are at the following link from the GitHub site:

https://github.com/debenavides/NDLP-Phishing/

The content of each code file in the shared repository is described below.

Preprocessed_dataset.csv

This dataset stores the 9761 valid rows obtained, of which 8589 are phishing rows and 1172 are from ham. This dataset contains three columns: context, which contains the text obtained from each web page, and the categorical columns, phishing and ham, which indicate whether it is a phishing web page or not. This data is ready to be entered into the NLP and DL algorithm.

NDLP Phishing Tuned Model.ipynb

This file contains the model already tuned with optimized parameters. In this model, the variations of all the hyper-parameters analyzed have been made. The experiment was performed on the Preprocessed_dataset.csv file data. Finally, the already trained model NDLP_model.h5 is generated and saved in the model.

NDLP_model.h5

This file is the saved model of the NDLP Phishing Tuned Model.ipynb, which is then sent to production to detect a new phishing page.

One Web Page analysis.ipynb

This code contains two parts. In the first part, the HTML text obtained from a new web page is cleaned, and in the second part, a prediction is made, calling the trained model NDLP_model.h5.

Ext_NDLP.zip

This folder contains all the Chrome extension files. Before running this extension, it must be added to the Chrome browser. This folder includes the code to pre-process the web page and the fine-tuned NDLP_model to analyze the pre-processed text.

7.2.2. Google Chrome extension

Figure 7.3 shows how the developed extension works internally. The browser extension is configured using the manifest.json file, which acts as a configuration map defining es-

sential properties such as name, version, and required permissions. The popup interface, popup.html, provides a simple user interface with a button that, when activated by the popup.js script, sends a message to the content script content.js to start capturing text on the current web page. The content.js script runs in the web page context and uses the browser API to capture the page text using document.body.innerText. Then, the fetch function sends an HTTP request to the Flask server with the captured text as JSON data.

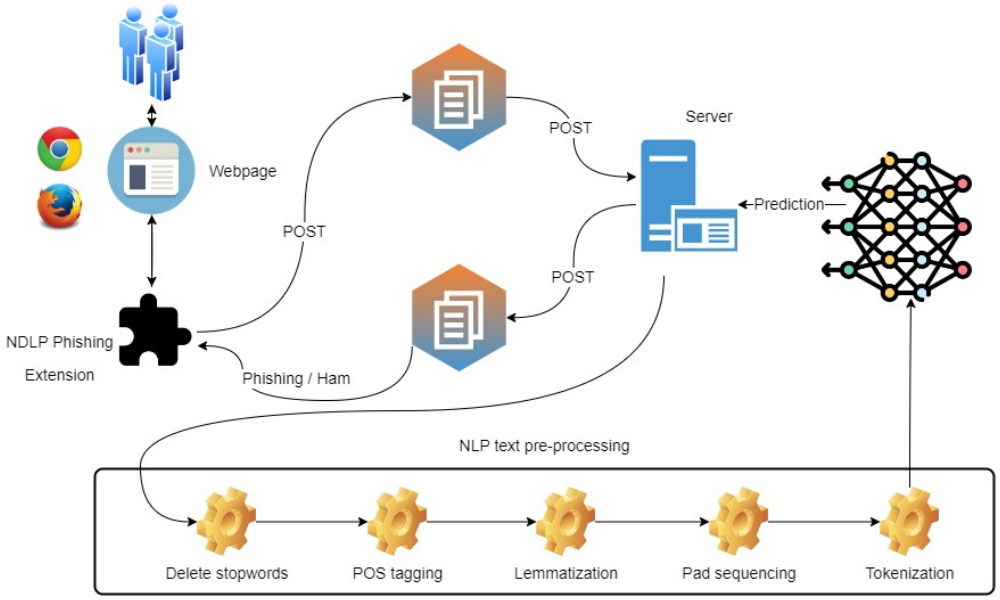


Figure 7.3: Design of the proposed application NDLP phishing

On the server side, Flask is used. Flask is a server and an environment to load and run trained DL models. The deploy.py script configures a Flask server and defines routes, including the POST /processText path. When it receives a request on this route, it invokes the process_text function, which performs processing operations on the text. The process_text function is responsible for processing the received text and performing specific operations such as cleaning, tokenization, and 200-word selection. Once the above process is complete, a phishing DL model NDLP Phishing.h5 will be used to make the prediction. Once the Flask server has processed the text and made the prediction, it sends the result back to the client. The content.js script on the client side receives the server's response, including the prediction result. Based on the prediction result, the script displays an alert to the user in the browser interface. This comprehensive interaction flow between the client and the server ensures that the browser extension can capture, process, and evaluate the text of the current web page, allowing a response to potential phishing attempts.

7.2.3. NDLP Execution

To use this application, we open the web page to be analyzed in the Google Chrome browser, then click the extension icon shown in Figure 7.4. The interface in Figure 7.5 will appear.

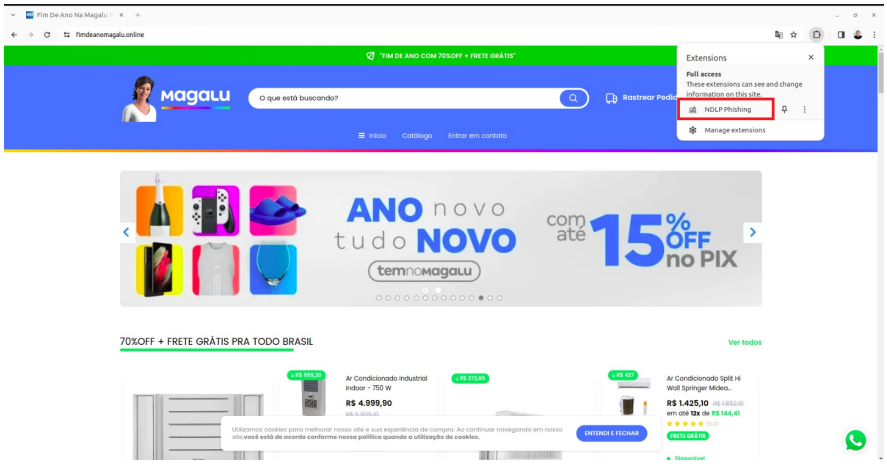


Figure 7.4: NDLP extension icon

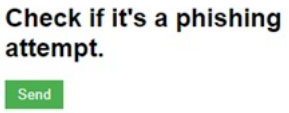


Figure 7.5: NDLP Message to check if the web page is a phishing attack attempt

When the user clicks on the interface in Figure 7.5, the NDLP program is executed, which analyzes the text of the web page and reports in Figure 7.6 and Figure 7.7 the percentage of probability that the page under analysis is phishing or not.

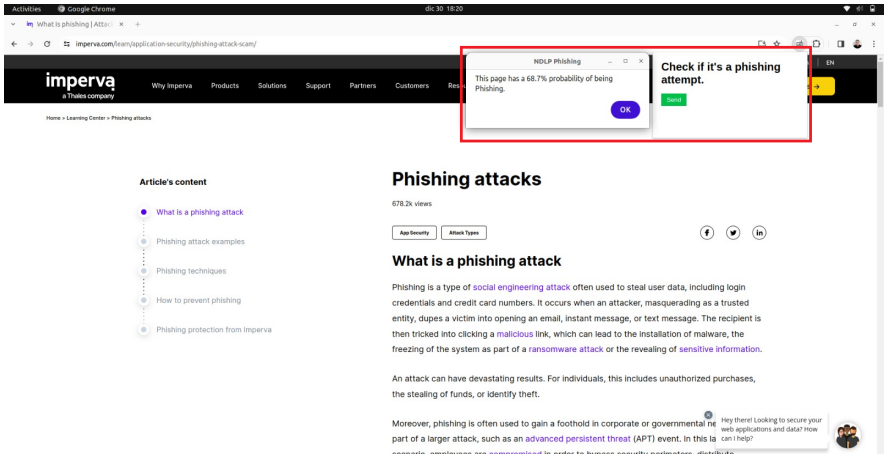


Figure 7.6: NDLP execution

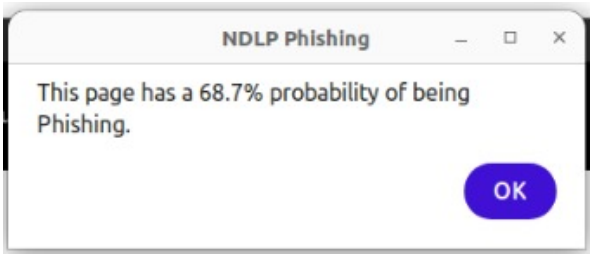


Figure 7.7: Message showing the percentage of a web page being phishing or not

7.2.4. Discussion

Although articles [130], [131], [132], [133], [134], [135], [136], [137], and [138] address the problem of detecting a Phishing attack through DL and NLP, none develops an application as in our proposal. In addition, articles that detect Phishing attacks using DL and NLP on web pages were also searched, but only the article [2] was found. In contrast, the others were oriented only to Phishing emails or Social Networks, leaving much to be investigated in the content of web pages. For this reason, the model proposed by the article [2] was taken as the starting point of our study, which detects phishing attacks using DL and NLP on the text contained in web pages. To compare the starting algorithm of the [2] article model with our proposed tuned algorithm, NDLP, we recreated the experiment of [2] on a personal computer. We ran it with our proposed tuned algorithm. (https://github.com/debenavides/NDLP_Comparative). Table 12 shows how our proposal configures the hyper-parameters of [2] and our proposal.

Table 7.9: Tuned hyper-parameters for the NDLP application.

	BiGRU layers	**Dropout**	**Batch size**	**Epochs**	**BiGRU neurons**	**GloVe Dimension**	**Elapsed time (min)**	**K-fold mean accuracy**
[2]	1	0.1	128	10	128	100	5h 32min	0.9682
Our tuned NDLP model	1	0.1	64	5	64	50	0h 26min	0.9855

After running the experiment, Table 7.9 shows that in the model proposed by [2], a 96.82% mean accuracy was obtained. In comparison with our tuned proposal, a 0.9855 mean accuracy was obtained. Thus, our tuned algorithm exceeds 1.7% of the algorithm proposed in [2] in a processing time of 12 times less.

Figure 7.8 shows the behavior of the two algorithms: the non-tuned one with ten epochs and the tuned one with five epochs. Also, in Figure 7.8, it can be observed that the non-tuned algorithm starts to fall after epoch 7, reaching even below 99.86%, while the tuned algorithm reaches epoch five up to 99.90%.

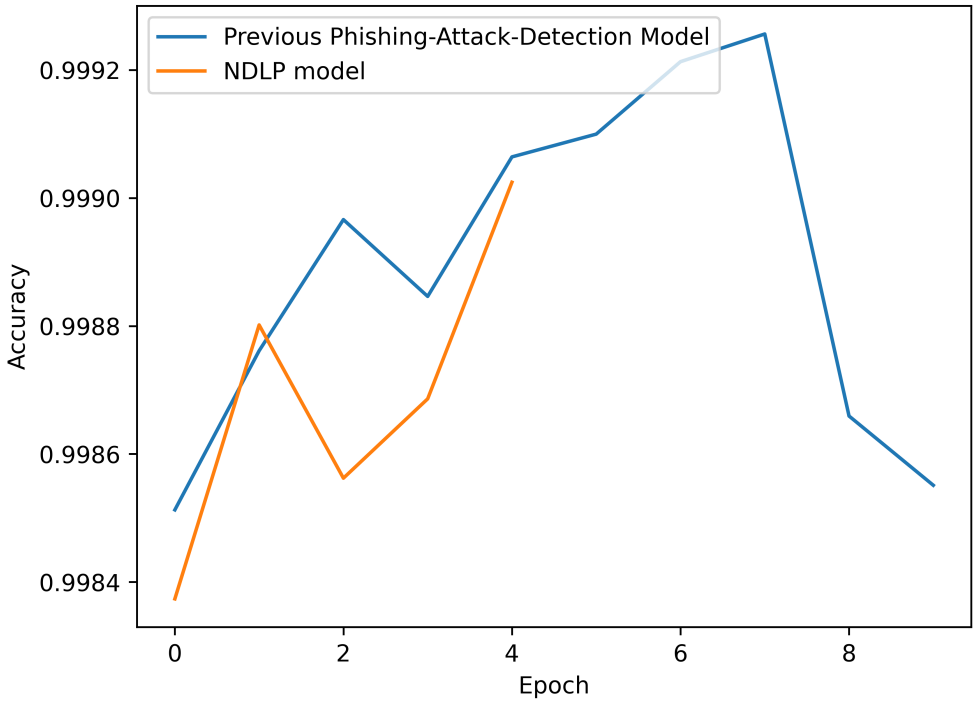


Figure 7.8: Comparison between the accuracy of the algorithms of the proposed model [2] and our refined NDLP

Table 7.9 shows that after our experiments, the hyper-parameters that did not change based on those initially had in the [2] model were BiGRU layers and Dropout. Also, Table 7.9 shows a decrease in the hyper-parameters batch size, epochs, BiGRU neurons, and GloVe dimensions, established at 0.9855. Thus, it can be determined that the values obtained with the tuned hyper-parameters only decrease the mean accuracy of the proposal from [2] to our proposal by 1%. On the other hand, the training time with our proposal is four times faster than the proposal [2]. Therefore, we have accepted the hyper-parameters displayed in Table 7.8 for our model as the foundation for the application developed as part of this research. With our proposal, we intend to fill the research gap by creating an application capable of detecting phishing attacks on web pages using the latest State-of-the-art technologies, DL, and NLP-GloVe. In addition, our application can capture only the text content of any web page, preprocessing it with NLP-GloVe and analyzing it with DL, predicting with a mean accuracy of over 99.00% whether a page is not phishing. The prediction is made using the previously obtained NDLP Phishing.h5 model; this application was installed through a Google Chrome extension and it is very intuitive.

7.3. Chapter Summary

In Chapter 7, it was determined that the algorithm that best fits our model is the BiGRU deep learning algorithm. However, in that chapter, they did not adjust the hyper-parameters to obtain an application that consumes little computational expense and a better mean accuracy. Due to this, in the first section of this chapter, the hyper-parameter adjustment is carried out to obtain a robust and lightweight model. The reproduction of the experiment can be carried out based on the code shared in the link: https://github.com/debenavides/NDLP-hyper-parameter-tuning/.

On the other hand, in the second section of this chapter, an extension is designed and implemented that uses the developed model to detect phishing attacks on the text of web pages. This extension is called NDLP phishing (to refer to the fact that NLP and DL are used), and it detects these phishing attacks. The reproduction application's source code can be downloaded from the link: https://github.com/debenavides/NDLP-Phishing/.

Chapter 8

Conclusions and Future Work

Contents

8.1. Conclusions

Phishing is a Social Engineering cyber attack in which attackers obtain confidential information from end users, mainly to defraud them. There are hardware, software, and awareness measures to confront this type of attack, but it is enough for an end user to fall for the deception and thus open and enter their information through a malicious web page. Therefore, the main objective of this thesis was to develop a model to help end users detect Phishing attacks in a relatively simple but highly accurate way.

Thus, this research proposes an innovative phishing-detection model to detect Phishing attacks using Deep Learning and Natural Processing Language. For this, first, we conducted a literature review. Then, we evaluated the personality traits and behaviors of victims of this type of attack. After that, we determined what types of data from the content of the web pages, HTML code or clear text, are more accurate for detecting phishing attacks. In the main part of our work, a detection model was proposed to determine which Deep Learning algorithm is most effective for phishing detection. Finally, the algorithm was fine-tuned and implemented as a Chrome browser extension.

This research used Barbara Kithchham's systematic literature review methodology to carry out the literature review [38]. According to the literature review, there still is a research gap in web page text analysis using Deep Learning and Natural Language Processing. Al-

though we found several articles on combating phishing with NLP and DL, most were oriented toward combating phishing emails, not web pages. As for tackling web pages, most studies were oriented toward working on URLs and not on the text of the web pages. Some articles were found that used Deep Learning to detect these attacks on the text of web pages. Still, however, they did not use Natural Language Processing to pre-process the text entered into the algorithms and thus take advantage of its rich semantics. The emphasis on web pages was predominantly on URL analysis, neglecting the text within the web pages. Furthermore, we found no evidence of these models being implemented in web browsers, highlighting the novelty of our approach.

A preliminary study was presented in *A Comparative Study of Deep Learning Algorithms in the Detection of Phishing Attacks Based on HTML and Text Obtained from Web Pages*. In this study, it was determined that if we execute the Deep Learning algorithm either with HTML or with text, it does not produce a greater difference in accuracy in the application of an algorithm. For this experiment, Deep Neural Network, Recurrent Neural Network, Convolutional Neural Network, and Recurrent Convolutional Neural Network algorithms were executed on the content of a large text dataset with HTML code and text. The average of the metrics obtained with HTML was 85%, and the overall metrics obtained with text averaged 84%.

The main objective of this work was to create a model for detecting phishing attacks using Deep Learning (DL) and Natural Language Processing (NLP). We made the phishing-detection model using the Keras Embedding Layer with the GloVe dictionary to take advantage of the semantic and syntactic features of the web page text. Our proposed model used automatic features of the text contained in web pages, applying word-level embedding methods to represent these features in vector form. These vectors were then input to the LSTM, BiLSTM, GRU, and BiGRU algorithms to identify phishing pages. After the experiment, it can be determined that BiGRU was the DL algorithm that performed better with the data previously analyzed with NLP than the other three DL algorithms, LSTM, BiLSTM, and GRU. On the other hand, BiLSTM gave the worst results, even below expected. The validity of our model was demonstrated using the K-fold cross-validation technique, which yielded a mean accuracy of LSTM 96.71%, BiLSTM 97.20%, GRU 97.29%, and the best, BiGRU 97.39%.

After implementing the model, our objective was to fine-tune it. Thus, we tested the model with three text lengths L obtained from web pages: L = 200, L = 500, and L = 1000; however, the results were similar, so it was determined that 200 was the optimal size for this

analysis. To fine-tune our algorithm, we tested different values of BiGRU layers, dropout, batch size, epochs, neurons per layer, and GloVe dictionary dimension. Hence, we refined the initial model based on the text of the web pages, and 98.55% of mean accuracy was obtained, with batch size = 64, five epochs, 64 neurons, and GloVe dimension = 50. That is 1.7% more than the initial solution proposed in [2] and 12 times faster for training and test response.

The complementary objective of this research was to develop an end-user-friendly application that can efficiently detect phishing attacks using NLP and DL on the text contained in the body of web pages to take advantage of the semantic and syntactic content of the text. To do this, we have developed a user-friendly extension for Google Chrome. This extension, built on the base of our refined model, not only detects phishing attacks by analyzing the text of web pages using NLP and applying the BiGRU algorithm but also ensures a secure browsing experience. It does this without introducing technical complexities, making it easy for all users, regardless of their technical proficiency. This seamless integration of advanced technology into everyday browsing is a significant step towards safer Internet use.

Also, to highlight the justification of this study, we made two works, shown in the annexes. First, in the study *A Framework Based on Personality Traits to Identify Vulnerabilities to Social Engineering Attacks*, it was determined that the end users who have the most significant tendency to be victims of phishing attacks are those who have the following traits: Openness (28.8%), Agreeableness (27.9%), Conscientiousness (20.2%), Neuroticism (11.5%), and Extraversion (11.5%). The Five-Factor Personality Model [139] was used to determine each respondent's personality. Next, in *Analysis of vulnerabilities associated with Social Engineering attacks based on user behavior*, a study was carried out at a higher education institution, specifically on teachers, administrative staff, and student population. It was determined that the group of students is more prone to receive attacks from Social Engineering due to their overconfidence and lack of experience. On the contrary, teachers and administrative staff are least likely to receive attacks from Social Engineering when handling confidential information and being aware of the dangers of losing any information. The four risk parameters of study [140] used to evaluate each respondent's risk where risk behavior, conservative behavior, exposure to offense, and perception of risk.

A significant limitation in the timely delivery of this work was the requirement for scientific publications, which either took a long time to be accepted and published or were never reviewed by the journals. In the particular case of this research, in one year, there has yet to be a response from one of the manuscripts. Finally, when experimenting with large amounts

of data, Deep Learning algorithms, and natural processing language, it is necessary to have robust hardware equipment. In our case, the RIG server used in this work was between twelve and forty times faster than a personal computer.

8.2. Future Work

This research was based on refining the BiGRU DL algorithm, which performs text analysis by NLP using embedding with the GloVe dictionary over text; however, initially developed for NLP tasks, Transformer algorithms have revolutionized the DL field. Thus, we plan to create an algorithm that detects phishing attacks based on the four most used Transformer algorithms. In this future work on applying various Transformer algorithms to detect phishing attacks, we will also make an extension that can be installed in addition to the Chrome browser in Mozilla Firefox, Edge, and Opera browsers.

We also plan to evaluate our model with other word embedding mechanisms, such as Fast Text, Word2Vec, or BERT, to evaluate their performance in the NLP domain. In addition, to improve the performance of DL algorithms, we plan to implement an attention-based DL architecture, which can differentiate which parts are more critical than others, depending on the context.

While in this research, we obtained acceptable results to face phishing attacks with NLP execution and DL algorithms, in the future, we plan to conduct a comparative study in which we can tune the parameters of both the embedding layer and DL algorithm layers for better results. An algorithm that detects these attacks is also required, using ensemble methods of at least two DL algorithms on various levels, such as URL, third-party information, and web content.

In future research, we plan to develop a model that considers users' personality traits and behavior to warn them when they are very likely to be the victim of a social engineering attack. This model will incorporate data from browsing history, text typed on social networks, and click and typing patterns.

Our model works well with the current training and test data and generalizes well for new attacks not yet seen; however, the continuous evolution of attacks requires the model to be constantly updated. For this reason, a module will be created to retrain the model with new attacks and update the application so that it does not lose its high accuracy.

The model was trained and evaluated with a single data set; however, different data sets are planned to evaluate the model. In addition, these new datasets will be merged

into a single dataset to retrain the model and re-measure its metrics. On the other hand, this work used unbalanced data, which is why k-fold cross-validation was used, resulting in an acceptable mean accuracy. Future experiments will use data augmentation and other techniques to balance the binary classification data and other more effective metrics for unbalanced data.

Bibliography

[1] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, and D. Nuñez-Agurto, "Ndlp phishing: A fine-tuned application to detect phishing attacks based on natural language processing and deep learning," *International Journal of Interactive Mobile Technologies*, vol. 18, no. 10, pp. 173–190, 2024.

[2] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, D. Nuñez-Agurto, and G. Rodríguez-Galán, "A phishing-attack-detection model using natural language processing and deep learning," *Applied Sciences*, vol. 13, no. 9, p. 5275, 2023.

[3] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, and M. Sanchez, "Classification of phishing attack solutions by employing deep learning techniques: A systematic literature review," *Smart Innovation, Systems and Technologies*, vol. 152, pp. 51–64, 2020, ISSN: 21903026. DOI: 10.1007/978-981-13-9155-2_5/COVER.

[4] E. Benavides-Astudillo, W. Fuertes, S. Sanchez-Gordon, G. Rodriguez-Galan, V. Martínez-Cepeda, and D. Nuñez-Agurto, "Comparative study of deep learning algorithms in the detection of phishing attacks based on html and text obtained from web pages," in *Applied Technologies: 4th International Conference, ICAT 2022, Quito, Ecuador, November 23–25, 2022, Revised Selected Papers, Part I*, Springer, 2023, pp. 386–398.

[5] E. Benavides-Astudillo, N. Tipan-Guerrero, G. Castillo-Zambrano, *et al.*, "A Framework Based on Personality Traits to Identify Vulnerabilities to Social Engineering Attacks," *Communications in Computer and Information Science*, vol. 1535 CCIS, pp. 381–394, 2022, ISSN: 18650937.

[6] E. Benavides-Astudillo, L. Silva-Ordoñez, R. Rocohano-Rámos, *et al.*, "Analysis of vulnerabilities associated with social engineering attacks based on user behavior," *Communications in Computer and Information Science*, vol. 1535, pp. 351–364, 2022, ISSN: 18650937. DOI: doi.org/10.1007/978-3-031-03884-6_26. [On-

line]. Available: https://link.springer.com/chapter/%5C%5C10.1007/978-3-031-03884-6%5C_26.

[7] L. R. Goldberg, “An alternative “description of personality”: The big-five factor structure,” in *Personality and Personality Disorders*, Routledge, 2013, pp. 34–47.

[8] K. D. Mitnick and W. L. Simon, *The art of deception: Controlling the human element of security*. John Wiley & Sons, 2003.

[9] E. Benavides-Astudillo, W. Fuertes-Díaz, and S. Sánchez-Gordon, “Un experimento para crear conciencia en las personas acerca de los ataques de ingeniería social,” *Revista Ciencia UNEMI*, vol. 13, no. 32, pp. 27–40, 2020.

[10] E. Benavides, W. Fuertes, S. Sanchez, and D. Nuñez-Agurto, *Caracterización de los ataques de phishing y técnicas para mitigarlos. ataques: Una revisión sistemática de la literatura. cienc. y tecnol. 13 (1), 97–104 (2020)*.

[11] B. Espinoza, J. Simba, W. Fuertes, E. Benavides, R. Andrade, and T. Toulkeridis, “Phishing attack detection: A solution based on the typical machine learning modeling cycle,” in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2019, pp. 202–207.

[12] D. Oña, L. Zapata, W. Fuertes, G. Rodríguez, E. Benavides, and T. Toulkeridis, “Phishing attacks: Detecting and preventing infected e-mails using machine learning methods,” in *2019 3rd Cyber Security in Networking Conference (CSNet)*, IEEE, 2019, pp. 161–163.

[13] G. Rodríguez, J. Torres, P. Flores, E. Benavides, and P. Proaño, “Trusted phishing: A model to teach computer security through the theft of cookies,” in *The International Conference on Advances in Emerging Trends and Technologies*, Springer, 2019, pp. 390–401.

[14] W. Fuertes, D. Arévalo, J. D. Castro, *et al.*, “Impact of social engineering attacks: A literature review,” *Developments and Advances in Defense and Security: Proceedings of MICRADS 2021*, pp. 25–35, 2022.

[15] B. B. Gupta, A. Tewari, A. K. Jain, and D. P. Agrawal, *Fighting against phishing attacks: state of the art and future challenges*, Dec. 2017. DOI: 10.1007/s00521-016-2275-y. [Online]. Available: http://link.springer.com/10.1007/s00521-016-2275-y.

[16] M. Jakobsson and S. Myers, *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley, 2007.

[17] A. K. Jain and B. Gupta, "Phish-safe: Url features-based phishing detection system using machine learning," in *Cyber Security*, Springer, 2018, pp. 467–474.

[18] A. A. Akinyelu, "Machine learning and nature inspired based phishing detection: A literature survey," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 05, p. 1 930 002, 2019.

[19] M. Hiransha, N. A. Unnithan, R. Vinayakumar, K. Soman, and A. Verma, "Deep learning based phishing e-mail detection," in *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*, Tempe, AZ, USA, 2018.

[20] T. Phoka and P. Suthaphan, "Image based phishing detection using transfer learning," in *2019 11th International Conference on Knowledge and Smart Technology (KST)*, IEEE, 2019, pp. 232–237.

[21] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: An application of large-scale online learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 681–688.

[22] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 639–648.

[23] A. K. Jain and B. B. Gupta, "Phishing detection: Analysis of visual similarity based approaches," *Security and Communication Networks*, vol. 2017, 2017.

[24] B. Wei, R. A. Hamad, L. Yang, *et al.*, "A deep-learning-driven light-weight phishing detection sensor," *Sensors*, vol. 19, no. 19, p. 4258, 2019.

[25] Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang, "Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism," *IEEE Access*, vol. 7, pp. 56 329–56 340, 2019.

[26] X. Zhang, Y. Zeng, X.-B. Jin, Z.-W. Yan, and G.-G. Geng, "Boosting the phishing detection performance by semantic analysis," in *2017 ieee international conference on big data (big data)*, IEEE, 2017, pp. 1063–1070.

[27] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious url detection using machine learning: A survey," *arXiv preprint arXiv:1701.07179*, 2017.

[28] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran, “A lexical approach for classifying malicious urls,” in *2015 international conference on high performance computing & simulation (HPCS)*, IEEE, 2015, pp. 195–202.

[29] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[30] A. C. Bahnsen, E. C. Bohorquez, S. Villegas, J. Vargas, and F. A. González, “Classifying phishing urls using recurrent neural networks,” in *2017 APWG symposium on electronic crime research (eCrime)*, IEEE, 2017, pp. 1–8.

[31] T. M. Mitchell, “Artificial neural networks,” *Machine learning*, vol. 45, no. 81, p. 127, 1997.

[32] A. M. Turing, “Computing machinery and intelligence (1950),” *The Essential Turing: the Ideas That Gave Birth to the Computer Age*, pp. 433–464, 2012.

[33] S. Mahdavifar and A. A. Ghorbani, “Application of deep learning to cybersecurity: A survey,” *Neurocomputing*, vol. 347, pp. 149–176, 2019.

[34] S. Selvaganapathy, M. Nivaashini, and H. Natarajan, “Deep belief network based detection and categorization of malicious urls,” *Information Security Journal: A Global Perspective*, vol. 27, no. 3, pp. 145–161, 2018.

[35] S. Knezevic, A. Datta, and S. Ulloa, “Tolerance of selected weed species to broadcast flaming at different growth stages,” in *Proceedings of the 8th European Weed Research Society workshop on physical and cultural weed control, Zaragoza, Spain*, 2009, pp. 98–103.

[36] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[37] *Glove: Global vectors for word representation*, `https://nlp.stanford.edu/projects/glove/`, Accessed: 2023-02-15.

[38] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering – A systematic literature review,” *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009, ISSN: 0950-5849. DOI: 10.1016/J.INFSOF.2008.09.009.

[39] P. Yang, G. Zhao, and P. Zeng, “Phishing website detection based on multidimensional features driven by deep learning,” *IEEE Access*, vol. 7, pp. 15 196–15 209, 2019.

[40] R. Vinayakumar, K. Soman, and P. Poornachandran, “Evaluating deep learning approaches to characterize and classify malicious url’s,” *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1333–1343, 2018.

[41] A. Bartoli, A. De Lorenzo, E. Medvet, and F. Tarlao, “Personalized, browser-based visual phishing detection based on deep learning,” in *International Conference on Risks and Security of Internet and Systems*, Springer, 2018, pp. 80–85.

[42] J. Zhang and X. Li, “Phishing detection method based on borderline-smote deep belief network,” in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, Springer, 2017, pp. 45–53.

[43] W. Wang, F. Zhang, X. Luo, and S. Zhang, “Pdrcnn: Precise phishing detection with recurrent convolutional neural networks,” *Security and Communication Networks*, vol. 2019, 2019.

[44] C. Sur, “Ensemble one-vs-all learning technique with emphatic & rehearsal training for phishing email classification using psychology,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 30, no. 6, pp. 733–762, 2018.

[45] W. Ali and A. A. Ahmed, “Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting,” *IET Information Security*, vol. 13, no. 6, pp. 659–669, 2019.

[46] M. Nguyen, T. Nguyen, and T. H. Nguyen, “A deep learning model with hierarchical lstms and supervised attention for anti-phishing,” *arXiv preprint arXiv:1805.01554*, 2018.

[47] X. X. Sun, S. Dai, and Y. X. Wang, “A platform for automatic identification of phishing urls in mobile text messages,” in *Journal of Physics: Conference Series*, IOP Publishing, 2018, p. 042 009.

[48] C. Coyotes, V. S. Mohan, J. Naveen, R. Vinayakumar, K. Soman, and A. Verma, “Ares: Automatic rogue email spotter,” in *Proc. 1st AntiPhishing Shared Pilot 4th ACM Int. Workshop Secur. Privacy Anal.(IWSPA)*, Tempe, AZ, USA, 2018.

[49] T. Hong, C. Choi, and J. Shin, “Cnn-based malicious user detection in social networks,” *Concurrency and Computation: Practice and Experience*, vol. 30, no. 2, e4163, 2018.

[50] P. Yi, Y. Guan, F. Zou, Y. Yao, W. Wang, and T. Zhu, “Web phishing detection using a deep learning framework,” *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[51] D. Aksu, Z. Turgut, S. Üstebay, and M. A. Aydin, “Phishing analysis of websites using classification techniques,” in *International Telecommunications Conference*, Springer, 2019, pp. 251–258.

[52] W. Chen, W. Zhang, and Y. Su, “Phishing detection research based on lstm recurrent neural network,” in *International Conference of Pioneering Computer Scientists, Engineers and Educators*, Springer, 2018, pp. 638–645.

[53] G. K. Soon, L. C. Chiang, C. K. On, N. M. Rusli, and T. S. Fun, “Comparison of ensemble simple feedforward neural network and deep learning neural network on phishing detection,” in *Computational Science and Technology*, Springer, 2020, pp. 595–604.

[54] G. Vrbančič, I. Fister Jr, and V. Podgorelec, “Parameter setting for deep neural networks using swarm intelligence on phishing websites classification,” *International Journal on Artificial Intelligence Tools*, vol. 28, no. 06, p. 1 960 008, 2019.

[55] S. Douzi, M. Amar, and B. El Ouahidi, “Advanced phishing filter using autoencoder and denoising autoencoder,” in *Proceedings of the International Conference on Big Data and Internet of Thing*, 2017, pp. 125–129.

[56] I. Torroledo, L. D. Camacho, and A. C. Bahnsen, “Hunting malicious tls certificates with deep neural networks,” in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, 2018, pp. 64–73.

[57] G. Vrbančič, I. Fister Jr, and V. Podgorelec, “Swarm intelligence approaches for parameter setting of deep learning neural network: Case study on phishing websites classification,” in *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, 2018, pp. 1–8.

[58] S. S. Kovalluri, A. Ashok, and H. Singanamala, “Lstm based self-defending ai chatbot providing anti-phishing,” in *Proceedings of the First Workshop on Radical and Experiential Security*, 2018, pp. 49–56.

[59] S. Shivangi, P. Debnath, K. Saieevan, and D. Annapurna, “Chrome extension for malicious urls detection in social media applications using artificial neural networks and

long short term memory networks," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2018, pp. 1993–1997.

[60] K. Shima, D. Miyamoto, H. Abe, *et al.*, "Classification of url bitstreams using bag of bytes," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, 2018, pp. 1–5.

[61] A. Vazhayil, R. Vinayakumar, and K. Soman, "Comparative study of the detection of malicious urls using shallow and deep networks," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2018, pp. 1–6.

[62] W. Yao, Y. Ding, and X. Li, "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, 2018, pp. 645–650.

[63] J. Spaulding and A. Mohaisen, "Defending internet of things against malicious domain names using d-fens," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, IEEE, 2018, pp. 387–392.

[64] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Detecting homoglyph attacks with a siamese neural network," in *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2018, pp. 22–28.

[65] X. Zhang, D. Shi, H. Zhang, W. Liu, and R. Li, "Efficient detection of phishing attacks with hybrid neural networks," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, IEEE, 2018, pp. 844–848.

[66] V. Ra, B. G. HBa, A. K. Ma, S. KPa, P. Poornachandran, and A. Verma, "Deepantiphishnet: 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, 2018, pp. 1–11.

[67] H. Yan, X. Zhang, J. Xie, and C. Hu, "Detecting malicious urls using a deep learning approach based on stacked denoising autoencoder," in *Chinese Conference on Trusted Computing and Information Security*, Springer, 2018, pp. 372–388.

[68] K. Sumathi and V. Sujatha, "Deep learning based-phishing attack detection," *International Journal of Recent Technology and Engineering*, 2019.

[69] Y. Huang, Q. Yang, J. Qin, and W. Wen, “Phishing url detection via cnn and attention-based hierarchical rnn,” in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, 2019, pp. 112–119.

[70] M. Arivukarasi and A. Antonidoss, “A cognitive support for identifying phishing websites using bi-lstm and rnn,” *International Journal of Recent Technology and Engineering*, vol. 8, no. 2, pp. 3097–3102, Jul. 2019, ISSN: 22773878. DOI: 10.35940/ijrte.B2646.078219.

[71] J. Ya, T. Liu, P. Zhang, J. Shi, L. Guo, and Z. Gu, “Neuralas: Deep word-based spoofed urls detection against strong similar samples,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–7.

[72] M. Chatterjee and A.-S. Namin, “Detecting phishing websites through deep reinforcement learning,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, vol. 2, 2019, pp. 227–232.

[73] Y. Liang, J. Kang, Z. Yu, B. Guo, X. Zheng, and S. He, “Leverage temporal convolutional network for the representation learning of urls,” in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, 2019, pp. 74–79.

[74] R. Vinayakumar, K. Soman, P. Poornachandran, S. Akarsh, and M. Elhoseny, “Deep learning framework for cyber threat situational awareness based on email and url data analysis,” in *Cybersecurity and Secure Information Systems*, Springer, 2019, pp. 87–124.

[75] I. Ahmad, M. A. Parvez, and A. Iqbal, “Typowriter: A tool to prevent typosquatting,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, vol. 1, 2019, pp. 423–432.

[76] H. Yuan, X. Chen, Y. Li, Z. Yang, and W. Liu, “Detecting phishing websites and targets based on urls and webpage links,” in *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, 2018, pp. 3669–3674.

[77] G. Lingam, R. R. Rout, and D. Somayajulu, “Deep q-learning and particle swarm optimization for bot detection in online social networks,” in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2019, pp. 1–6.

[78] *PhishTank | Join the fight against phishing*. [Online]. Available: https://www.phishtank.com/ (visited on 05/05/2020).

[79] *OpenPhish - Phishing Intelligence*. [Online]. Available: https://openphish.com/ (visited on 04/04/2020).

[80] *MDL*. [Online]. Available: https://www.malwaredomainlist.com/ (visited on 05/05/2020).

[81] *Phishload - Download*. [Online]. Available: https://www.medien.ifi.lmu.de/team/max.maurer/files/phishload/download.html (visited on 05/05/2020).

[82] *PhishLabs*. [Online]. Available: https://www.phishlabs.com/ (visited on 05/05/2020).

[83] *MalwareURL*. [Online]. Available: https://www.malwareurl.com/ (visited on 05/05/2020).

[84] *Phishing scams and spoof emails at MillerSmiles.co.uk*. [Online]. Available: http://www.millersmiles.co.uk/ (visited on 05/05/2020).

[85] *Keyword Research, Competitive Analysis, & Website Ranking | Alexa*. [Online]. Available: https://www.alexa.com/ (visited on 05/05/2020).

[86] *The Directory of the Web*. [Online]. Available: http://dmoztools.net/ (visited on 05/05/2020).

[87] *UCI Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/index.php (visited on 05/05/2020).

[88] *Common Crawl*. [Online]. Available: https://commoncrawl.org/ (visited on 02/11/2020).

[89] *360º websites*. [Online]. Available: https://www.awwwards.com/websites/360/ (visited on 02/11/2020).

[90] *Starting Point Directory*. [Online]. Available: http://www.stpt.com/directory/ (visited on 03/10/2020).

[91] *Yahoo! Directory*. [Online]. Available: https://web.archive.org/web/20141122194515/https://dir.yahoo.com/ (visited on 03/10/2020).

[92] J. Peck, C. Nie, R. Sivaguru, *et al.*, "Charbot: A simple and effective method for evading dga classifiers," *IEEE Access*, vol. 7, pp. 91 759–91 771, 2019.

[93] S. Salloum, T. Gaber, S. Vadera, and K. Sharan, "A systematic literature review on phishing email detection using natural language processing techniques," *IEEE Access*, 2022.

[94] R. Zieni, L. Massari, and M. C. Calzarossa, "Phishing or not phishing? a survey on the detection of phishing websites," *IEEE Access*, 2023.

[95] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Computing and Applications*, vol. 31, pp. 3851–3873, 2019.

[96] S. Marchal, G. Armano, T. Gröndahl, K. Saari, N. Singh, and N. Asokan, "Off-the-hook: An efficient and usable client-side phishing prevention application," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1717–1733, 2017.

[97] A. K. Jain and B. B. Gupta, "A machine learning based approach for phishing detection using hyperlinks information," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 2015–2028, 2019.

[98] S. Yen, M. Moh, and T.-S. Moh, "Detecting compromised social network accounts using deep learning for behavior and text analyses," *International Journal of Cloud Applications and Computing (IJCAC)*, vol. 11, no. 2, pp. 97–109, 2021.

[99] A. Ozcan, C. Catal, E. Donmez, and B. Senturk, "A hybrid DNN–LSTM model for detecting phishing URLs," *Neural Computing and Applications*, pp. 1–17, 2021, ISSN: 14333058. DOI: 10.1007/S00521-021-06401-Z/TABLES/7.

[100] X. Xiao, W. Xiao, D. Zhang, *et al.*, "Phishing websites detection via CNN and multi-head self-attention on imbalanced datasets," *Computers & Security*, vol. 108, p. 102 372, 2021, ISSN: 0167-4048. DOI: 10.1016/J.COSE.2021.102372.

[101] S. S. Sirigineedi, J. Soni, and H. Upadhyay, "Learning-based models to detect runtime phishing activities using URLs," *ACM International Conference Proceeding Series*, pp. 102–106, 2020. DOI: 10.1145/3388142.3388170.

[102] M. A. Adebowale, K. T. Lwin, and M. A. Hossain, "Intelligent phishing detection scheme using deep learning algorithms," *Journal of Enterprise Information Management*, 2020.

[103] A. Villanueva, C. Atibagos, J. De Guzman, J. C. D. Cruz, M. Rosales, and R. Francisco, "Application of natural language processing for phishing detection using machine and deep learning models," in *2022 International Conference on ICT for Smart Society (ICISS)*, IEEE, 2022, pp. 01–06.

[104] Z. Alshingiti, R. Alaqel, J. Al-Muhtadi, Q. E. U. Haq, K. Saleem, and M. H. Faheem, "A deep learning-based phishing detection system using cnn, lstm, and lstm-cnn," *Electronics*, vol. 12, no. 1, p. 232, 2023.

[105] S. Ariyadasa, S. Fernando, and S. Fernando, "Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using url and html," *IEEE Access*, vol. 10, pp. 82 355–82 375, 2022.

[106] N. Q. Do, A. Selamat, O. Krejcar, T. Yokoi, and H. Fujita, "Phishing webpage classification via deep learning-based algorithms: An empirical study," *Applied Sciences*, vol. 11, no. 19, p. 9210, 2021.

[107] P. K. Loh, A. Z. Lee, and V. Balachandran, "Towards a hybrid security framework for phishing awareness education and defense," *Future Internet*, vol. 16, no. 3, p. 86, 2024.

[108] I. S. Mambina, J. D. Ndibwile, D. Uwimpuhwe, and K. F. Michael, "Uncovering sms spam in swahili text using deep learning approaches," *IEEE Access*, 2024.

[109] A. Ejaz, A. N. Mian, and S. Manzoor, "Life-long phishing attack detection using continual learning," *Scientific Reports*, vol. 13, no. 1, p. 11 488, 2023.

[110] U. A. Butt, R. Amin, H. Aldabbas, S. Mohan, B. Alouffi, and A. Ahmadian, "Cloud-based email phishing attack using machine and deep learning algorithm," *Complex & Intelligent Systems*, vol. 9, no. 3, pp. 3043–3070, 2023.

[111] E. A. Aldakheel, M. Zakariah, G. A. Gashgari, F. A. Almarshad, and A. I. Alzahrani, "A deep learning-based innovative technique for phishing detection in modern security with uniform resource locators," *Sensors*, vol. 23, no. 9, p. 4403, 2023.

[112] M. K. Moussavou Boussougou and D.-J. Park, "Attention-based 1d cnn-bilstm hybrid model enhanced with fasttext word embedding for korean voice phishing detection," *Mathematics*, vol. 11, no. 14, p. 3217, 2023.

[113] S.-G. Nam, Y. Jang, D.-G. Lee, and Y.-S. Seo, "Hybrid features by combining visual and text information to improve spam filtering performance," *Electronics*, vol. 11, no. 13, p. 2053, 2022.

[114] A. Aljofey, Q. Jiang, A. Rasool, *et al.*, "An effective detection approach for phishing websites using url and html features," *Scientific Reports*, vol. 12, no. 1, p. 8842, 2022.

[115] M.-E. Maurer, *Phishload*, `https://www.medien.ifi.lmu.de/team/max.maurer/files/phishload/`, Accessed: 2022-11-16.

[116] M. Siddahartha, *Malicious URLs dataset*. [Online]. Available: `https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset`.

[117] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[118] A. A. Alsufyani and S. Alzahrani, *Social engineering attack detection using machine learning: Text phishing attack*, 2021.

[119] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, 2020, ISSN: 0167-2789. DOI: 10.1016/J.PHYSD.2019.132306. arXiv: 1808.03314.

[120] H. Deng, L. Zhang, and X. Shu, "Feature memory-based deep recurrent neural network for language modeling," *Applied Soft Computing*, vol. 68, pp. 432–446, 2018, ISSN: 1568-4946. DOI: 10.1016/J.ASOC.2018.03.040.

[121] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 8 1997, ISSN: 08997667. DOI: 10.1162/NECO.1997.9.8.1735.

[122] PapersWithCode, *Bidirectional lstm*, https://paperswithcode.com/method/bilstm, Accessed: 2023-01-21, (accessed: 21.01.2023).

[123] D. Britz, *Recurrent neural network tutorial, part 4 – implementing a gru and lstm rnn with python and theano*, https://dennybritz.com/posts/wildml/recurrent-neural-networks-tutorial-part-4/, Accessed: 2023-01-29, (accessed: 29.01.2023).

[124] Y. Deng, H. Jia, P. Li, X. Tong, X. Qiu, and F. Li, "A deep learning methodology based on bidirectional gated recurrent unit for wind power prediction," *Proceedings of the 14th IEEE Conference on Industrial Electronics and Applications, ICIEA 2019*, pp. 591–595, 2019. DOI: 10.1109/ICIEA.2019.8834205.

[125] X. Luo, W. Zhou, W. Wang, Y. Zhu, and J. Deng, "Attention-based relation extraction with bidirectional gated recurrent unit and highway network in the analysis of geological data," *IEEE Access*, vol. 6, pp. 5705–5715, 2018, ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2785229.

[126] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the internet of things networks of smart cities," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 4436–4456, 2020.

[127] E. Lee, *A gentle introduction to hyperparameter tuning with scikit learn and python*, https://drlee.io/a-gentle-introduction-to-hyperparameter-tuning-with-scikit-learn-and-python-835139d55ef, 2023.

[128] Y. Bergstra James; Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[129] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[130] M. Nguyen, T. Nguyen, and T. H. Nguyen, “A deep learning model with hierarchical lstms and supervised attention for anti-phishing,” *arXiv preprint arXiv:1805.01554*, 2018.

[131] S. Bagui, D. Nandi, S. Bagui, and R. J. White, “Machine learning and deep learning for phishing email classification using one-hot encoding,” *Journal of Computer Science*, vol. 17, no. 7, pp. 610–623, 2021.

[132] Q. Yaseen *et al.*, “Spam email detection using deep learning techniques,” *Procedia Computer Science*, vol. 184, pp. 853–858, 2021.

[133] A. Alhogail and A. Alsabih, “Applying machine learning and natural language processing to detect phishing email,” *Computers & Security*, vol. 110, p. 102 414, 2021.

[134] R. Brindha, S. Nandagopal, H. Azath, V. Sathana, G. P. Joshi, and S. W. Kim, “Intelligent deep learning based cybersecurity phishing email detection and classification.,” *Computers, Materials & Continua*, vol. 74, no. 3, 2023.

[135] M. Dewis and T. Viana, “Phish responder: A hybrid machine learning approach to detect phishing and spam emails,” *Applied System Innovation*, vol. 5, no. 4, p. 73, 2022.

[136] K. Haynes, H. Shirazi, and I. Ray, “Lightweight url-based phishing detection using natural language processing transformers for mobile devices,” *Procedia Computer Science*, vol. 191, pp. 127–134, 2021.

[137] S. Atawneh and H. Aljehani, “Phishing email detection model using deep learning,” *Electronics*, vol. 12, no. 20, p. 4261, 2023.

[138] J. E. Coyac-Torres, G. Sidorov, E. Aguirre-Anaya, and G. Hernández-Oregón, “Cyberattack detection in social network messages based on convolutional neural networks and nlp techniques,” *Machine Learning and Knowledge Extraction*, vol. 5, no. 3, pp. 1132–1148, 2023.

[139] P. T. Costa Jr and R. R. McCrae, “The five-factor model of personality and its relevance to personality disorders,” *Journal of personality disorders*, vol. 6, no. 4, pp. 343–359, 1992.

[140] G. Öğütçü, Ö. M. Testik, and O. Chouseinoglou, “Analysis of personal information security behavior and awareness,” *Computers & Security*, vol. 56, pp. 83–93, 2016.

[141] O. P. John, R. W. Robins, and L. A. Pervin, *Handbook of personality: Theory and research*. Guilford Press, 2010.

[142] D. Papatsaroucha, Y. Nikoloudakis, I. Kefaloukos, E. Pallis, and E. K. Markakis, “A survey on human and personality vulnerability assessment in cyber-security: Challenges, approaches, and open issues,” *arXiv preprint arXiv:2106.09986*, 2021.

Appendix A

Phishing, Personality Traits, and User Behavior

A.1. Personality Traits Vulnerable to Social Engineering Attacks

Some personalities are more prone to attacks than others because the attackers take advantage of the personality traits of their victims, such as friendliness, ignorance of basic security measures, naivety, or overconfidence. In this section, we propose a Framework based on personality traits, allowing us to know which people are more vulnerable than others. Faced with this scenario, the main aim of this study is to develop a tool that will determine which people are most vulnerable to social engineering attacks. The methodological process consisted of first determining the most common traits of users related to vulnerability. Then, the personality traits of a surveyed group were determined. Finally, the most vulnerable traits matched the respondents' personality traits. The personality traits in which our research was framed are known as the Five-Factor Model of personality. The methodology used for this study is explained in Figure A.1.

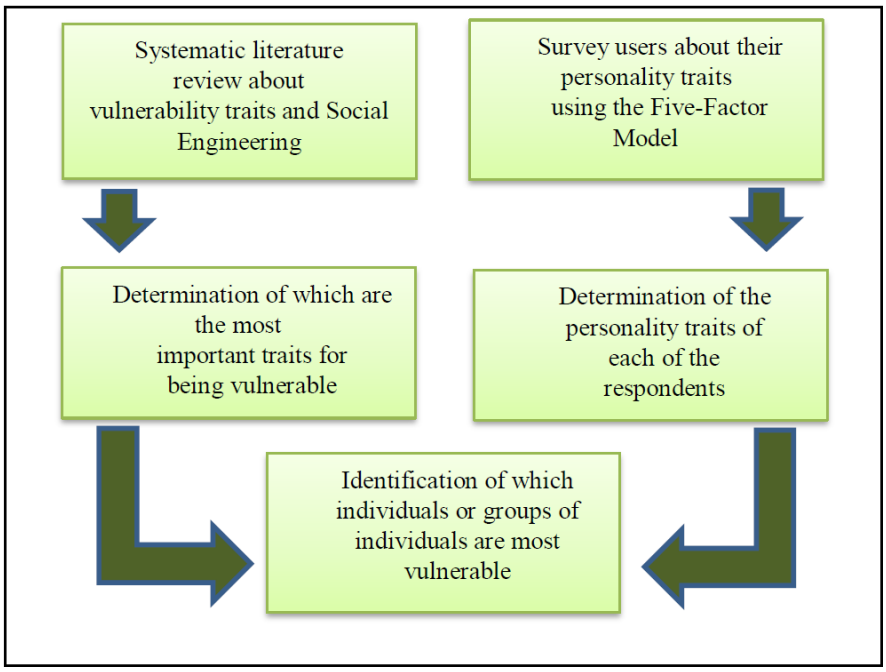


Figure A.1: Model for defining user vulnerability by personality trait.

A.1.1. Determination of the Most Important Personality Traits to be Vulnerable

Once the 36 publications were obtained as a result of the literature review of this research, the total reading of each was carried out. Hence, we extracted the main personality traits evaluated in these articles. Then, these personality traits were framed within the characteristics of the personality of the Five-Factor Model (FFM). They were also assigned colors to identify each personality trait, high or low. See A.2. At this point, we decided to give a high or low scale to each FFM property because, based on our study, a person with high openness is much more at risk than a person without openness.

Five-Factor Model	Abbreviation
Openness (High)	Op_H
Openness (Low)	Op_L
Conscientiousness (High)	Co_H
Conscientiousness (Low)	Co_L
Extraversion (High)	Ex_H
Extraversion (Low)	Ex_L
Agreeableness (High)	Ag_H
Agreeableness (Low)	Ag_L
Neuroticism (High)	Ne_H
Neuroticism (Low)	Ne_L

Figure A.2: Five-Factor Model with high and low scales and colors.

A.1.2. Surveying to Determine Personality Traits

Parallel to the previous steps, 146 people were surveyed to determine their personality traits, considering the Five-Factor personality model. Table A.1 lists the survey questions.

Table A.1: Distribution of survey questions for each FFM factor of personality.

1. Do you like to talk a lot?	22. Are you generally confident?
2. Do you tend to find fault with others?	23. Do you tend to be lazy?
3. Do you do a thorough job?	24. Are you emotionally stable?
4. Are you depressed or sad?	25. Are you creative and invent new things?
5. It is original; do you come up with new ideas?	26. Are you assertive, self-confident?
6. Is it reserved?	27. Can you be cold and distant?
7. Are you helpful and disinterested in others?	28. Do you persevere until you finish a task?
8. Can it be somewhat sloppy?	29. Can you be moody?
9. Are you relaxed, or do you handle stress well?	30. Do you value artistic and aesthetic experiences?
10. Are you curious about many different things?	31. Are you sometimes shy or inhibited?
11. Are you full of energy?	32. Are you considerate and kind to almost everyone?
12. Do you fight or argue with others?	33. Do you do things efficiently?
13. Are you a reliable worker?	34. Do you stay calm in stressful situations?
14. Are you usually tense?	35. Do you prefer routine work?
15. Are you resourceful and a deep thinker?	36. Are you sociable?
16. Do you have a lot of enthusiasm?	37. Are you sometimes rude to others?
17. Do you have a compassionate nature towards others?	38. Do you make plans and follow them?
18. Do you tend to be disorganized?	39. Do you get nervous easily?
19. Do you worry a lot?	40. Do you like to reflect and play with ideas?
20. Do you have an active imagination?	41. Do you have few artistic interests?
21. Do you tend to be quiet?	42. Do you like to cooperate with others?

A.1.3. Determination of the Personality Traits of Survey Respondents

After the survey, which 146 people from a higher education institution answered, the personality traits of each were determined. See Table A.2 with three examples of surveyed people.

Table A.2: Three examples of survey results to determine personality traits.

Respondent	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Openness
1	Low	Low	Low	Medium	Medium
2	Low	Medium	Medium	Low	High
3	Low	High	Medium	Low	High

A.1.4. High, Medium, and Low Risk in Each FFM Trait

According to [141] and [142], high, medium, and low values are established for each behavioral trait in FFM. See Table A.3. In addition, based on the results obtained from the interpretation of the analyzed publications, we have assigned them a red, yellow, and green signal according to the level of risk that this trait implies: high, medium, or low.

Table A.3: Traffic lights of high, medium, and low scales.

FFM	Low	Medium	High
Extraversion	People with low scores can be described as introverts; therefore, it is more difficult to get information from them.	People with average scores can be somewhat reserved but tend to generate conversation if their attention is captured.	High-scoring people are comfortable in large groups and tend to be enthusiastic, energetic, and highly talkative
Agreeableness	This type of person is usually critical, is not condescending and expresses hostility towards others.	They usually do not try to please anyone, but if necessary, they would be willing to help others, always being alert.	These people are likelier to help and trust others because they always assume the best in others.
Conscientiousness	In this trait, those are likelier to not stick to plans and follow their own rules without measuring consequences.	People with established values who can be trusted are not exempt from making small, careless mistakes.	Honesty, trust, strong self-orientation, and self-responsibility are the main characteristics of these people.
Neuroticism	These people tend to be more emotionally stable and better meditate on all their actions.	They tend to remain calm, but the more it forces them, the more quickly they reach their breaking point.	The higher someone's score on this trait, the more they tend to worry and make hasty decisions.
Openness	Favor conservative values. They are judges in conventional terms and uncomfortable with complexities. It could be wrong not to be aware of new attack methods.	People were willing to accept new experiences without losing conservative values.	They are open to new ideas and experiences and accept different beliefs.

A.1.5. Identification of Most Vulnerable Individuals or Groups

Once two things were obtained separately: (1) on the one hand, the essential characteristics in the literature framed in the FFM; (2) on the other, the personality traits of the respondents, also framed in the FFM; a match was performed to determine which people are the most vulnerable to Social Engineering attacks.

A.1.6. Survey Results to Determine Personality Traits of Users

Parallel to defining the most important personality traits, we surveyed university personnel to determine the respondents' personality traits. The sample taken is 146 out of the 800 university community members. This way, a score was determined for each respondent in each FFM factor. For instance, in Table A.4, we show the first nine results out of the 146 respondents.

Table A.4: Numerical results of the survey carried out framed in FFM.

Respondents	**Extraversion**	**Agreeableness**	**Conscientiousness**	**Neuroticism**	**Openness**
1	17	18	23	31	34
2	21	33	35	22	47
3	22	37	33	23	47
4	26	31	31	25	33
5	24	30	36	22	32
6	32	39	28	29	31
7	36	37	37	11	45
8	18	26	23	21	30
9	23	31	28	19	25

This study even determined that people with higher openness and agreeableness are likelier to be victims than those with low conscientiousness. In other words, determining low, medium, or high values in each FFM and each respondent can define whether this person is

more vulnerable than another person. Then, based on the values obtained from the survey of nine respondents in Table A.4, we created Table A.5, in which, as a traffic light, the risk of each respondent is indicated, highlighting in red the high-risk characteristics, in yellow those of medium risk and green those of low risk.

Table A.5: Traffic lights of the survey carried out framed in FFM.

Respondents	Extraversion	Agreeableness	Conscientiousness	Neuroticism	Openness
1	Low	Low	Low	Medium	Medium
2	Low	Medium	Medium	Low	High
3	Low	High	Medium	Low	High
4	Medium	Medium	Medium	Medium	Medium
5	Medium	Medium	High	Low	Medium
6	High	High	Medium	Medium	Medium
7	High	High	High	Low	High
8	Low	Low	Low	Low	Medium
9	Low	Medium	Medium	Low	Low

A.1.7. Determination of People Vulnerable to Social Engineering a Attacks

Table A.5 shows the example of nine people to whom the survey was carried out. It can be seen that the person who is most vulnerable to being a victim of a Phishing attack is the site person because he has three characteristics in red (Extraversion, Agreeableness, and Openness), while person five has four characteristics in green (Agreeableness, Conscientiousness, Neuroticism, and Openness). Following the previous example, the other respondents can be evaluated, too.

A.2. Vulnerable Behavior of People to Social Engineering Attacks

Certain personality traits, such as those studied in the previous section, cause certain behaviors that can lead us to be victims of a Phishing attack. Although our personality traits are intrinsic and we cannot control them, we can regulate people's behavior. By regulating this behavior, we can warn people before they commit an action that leads them to be a victim of a Phishing attack. In this section, a study is carried out on the behavior of people that makes them victims of attacks.

There are undoubtedly behaviors that make some people more exposed to Social Engineering attacks than others. For this reason, we conducted a study of teachers, administrators, and students in a higher education institution to determine the relationships between specific users and the Social Engineering attacks of which they are victims.

This study was carried out through a survey of 153 people, who, according to their responses, were evaluated to determine their level of the following behavioral Likert scales: Risky Behavior Scale (RBS), Conservative Behavior Scale (CBS), Exposure to Offence Scale (EOS), and Risk Perception Scale (RPS).

Next, the following four hypotheses were proposed:

- **H1**: There is no significant difference between the scales RBS, CBS, EOS, and RPS concerning their average.
- **H2**: There is no significant difference between the surveyed groups (teachers, administrators, and students) concerning their average.
- **H3**: The exposure to more hours/day that users have when using the Internet affects the average of the scales RBS, CBS, EOS, RPS.
- **H4**: There is a significant correlation between the averages of the scales RBS, CBS, EOS, and RPS.

A.2.1. RBS

This scale refers to the behavior of users in the face of risk towards Information Systems. The questions that were asked to the participants to know their score on this scale were the following:

- Do you use WhatsApp, Telegram, Messenger or similar chat programs?
- Do you use Meet, Teams, Zoom, or similar meeting programs?
- Do you use email?
- Do you use your Corporate or Institutional email address for your personal business?
- Do you use online banking?
- Do you make purchases or payments on the Internet?
- Do you use websites that provide services to citizens electronically (i.e., check identity
- number, payment of basic services, etc.)?
- Do you play video games online?
- Do you watch videos or movies online?
- When necessary, do you share your personal information on the Internet (i.e., first name, last name, date of birth, email, address, etc.)?
- Do you transfer confidential files on WhatsApp, Telegram or Messenger?
- Do you use online banking in places where there is access to the public Internet?
- Do you share your passwords with other people?
- Do you save your passwords by writing them in diaries or places that can be easily found?
- Do you open emails from strangers or download the attachments of those emails?

A.2.2. CBS

The objective of this scale is to measure the user's actions when using an Information System. The questions that were asked to the participants to know their score on this scale were the following:

- Do you use the original licensed software on your computer?
- Do you use programs like virus detection, spyware, etc?
- Do you delete temporary internet files and history before leaving a computer?

- Do you use long and complicated passwords that cannot be easily guessed for your Internet accounts and personal files?
- Do you use an electronic signature?
- Do you have a password to access your computer?
- Do you pay attention to the websites you visit, checking if they have the HTTPS lock in the address bar?
- Do you often change your passwords?
- Are you aware that others may use your personal information illegally?

A.2.3. EOS

This scale aims to measure the exposure that users have to any cybersecurity threat. The questions that were asked to the participants to know their score on this scale were the following:

- Have you had problems due to computer viruses?
- Have you experienced financial loss because of online shopping?
- Have you had problems sharing your personal information on the Internet?
- Have you received any notification of using your username and password on the Internet without your authorization?
- Have the files on your computer ever been stolen or deleted?
- Have you found fake accounts that use your confidential data or user profile?
- Do you use any entity that preserves your credit card details in online purchases, such as PayPal?

A.2.4. RPS

This scale measures the degree of risk or danger a user captures using information technologies. Of the following items, the participants were asked to indicate the degree of danger that they perceive:

- Computer virus.
- Lack of antivirus.
- Spyware (Keylogger, Screen logger, Trojan, etc.).
- File sharing programs (Google Drive, Dropbox, Mega, etc.).
- Chat programs (WhatsApp, Telegram, and Messenger.).
- Junk, spam, or junk email.
- Online games.
- USB or external memories.
- Macros in Microsoft Office applications (Word, Excel, etc.).
- Use of pirated programs.
- Download materials such as music, photos, or movies without paying anything.
- Open emails with advertising content.
- Use of online banking.
- Share information with strangers online.
- Online purchases.
- Using Wireless Wi-Fi.
- Downloading and using free or unlicensed programs.
- Delivery of identity card or driver's license number to security personnel at the entrance of a building.

A.2.5. Demographics

In addition, it was necessary to collect demographic information from the respondents. For this, the following questions were presented to the participants:

- Select your age range.
- Select your gender.

- Have you ever had training or experience in Internet Security?
- What is your average time of Internet use per day?
- Choose your occupation.
- What is your level of education?
- Choose the department, study, or work area to which you belong.
- How do you access the Internet from outside your workplace?
- Choose your department or study area.
- Choose the level you are at in your career.
- Enter your position within the Institution.

A.2.6. Results and Discussion

After completing the surveys, their responses were evaluated to answer the hypotheses raised. Below are the results obtained with their respective decisions:

Hypothesis H1: There is no significant difference between the scales RBS, CBS, EOS, and RPS concerning their average.

Regarding H1, it can be verified that there is a significant difference between the scales, which is why H1 is discarded. To contrast H1, we used the analysis of variance or ANOVA between scales. This ANOVA analysis allows us to identify the significant difference in the results obtained in the developed survey. To develop the ANOVA analysis between the scales RBS, CBS, EOS, RPS, the Significance Level α = 0.05 was used so that if the value of P (Probability) is less than that of α, H1 is rejected. Therefore, it is shown that there is a significant difference between the scales, as can be seen in Table A.6.

Table A.6: ANOVA calculation between scales.

Source of variations	Sum of Squares	Degree of Freedom	Mean Square Error	Error	Probability
Between groups	13.81	3	4.60	7.39	0.00
Within groups	28.63	46		0.62	
Total	42.44	49	0.87		

Table A.6 shows a significant difference between the scales analyzed with ANOVA. For this reason, it is necessary to complement the analysis by developing a Tukey test to identify

the scales that create this difference. Table A.7 shows the results of the scales compared to identify where the difference is created.

Table A.7: Tukey test between scales.

Group 1	Group 2	Media	Standard Error	P-value
RBS	CBS	0.15486	0,23244	0.965
RBS	EOS	0.9226	0.25280	0.061
RBS	RPS	0.66612	0.19167	0.080
CBS	EOS	0.76740	0.28113	0.229
CBS	RPS	0.82098	0.22774	0.065
EOS	RPS	1.58839	0.24849	0.000

Based on Table A.7 and the relationship between EOS and RPS, it can be observed that RPS positively influences EOS. In conclusion, users will be less exposed to Social Engineering attacks as they perceive risk better.

Hypothesis H2: There is no significant difference between the surveyed groups concerning their average.

In contrast to H1, in H2, the data were grouped between the three surveyed groups (i.e., teachers, administrators, and students) and the indicated scales. Table 13 shows the results obtained by ANOVA. By obtaining the value of P = 0.004 in the case of the CBS scale and of P = 0.021 in the case of the RPS scale, it is shown that there is a significant difference between the surveyed groups. According to the differences found, it was determined that the group of teachers has substantial differences between the group of administrative personnel and the group of students.

Table A.8: ANOVA calculation between academics, administrative staff, and students.

Scales						
RBS	Between groups	0.11	2	0.06	0.23	0.79
	Within groups	36.31	150	0.24	-	-
	Total	36.43	152	0.24	-	-
CBS	Between groups	2.53	2	1.26	5.61	0.004
	Within groups	33.79	150	0.23	-	-
	Total	36.32	152	0.24	-	-
EOS	Between groups	0.08	2	0.04	0.69	0.50
	Within groups	9.04	150	0.06	-	-
	Total	9.13	152	0.06	-	-
RPS	Between groups	1.83	2	0.91	3.94	0.021
	Within groups	34.78	150	0.23	-	-
	Total	36.61	152	0.24	-	-

In addition, the Tukey test was used between the scales and the groups to identify the group or groups that generate this difference. We find that the group of teachers generates the difference in both the CBS A.9 and the RPS scale A.10.

Table A.9: Tukey test between the groups (teachers, administrative staff, and students) and the CBS scale.

Group 1	Group 2	Media	Standard Error	P-value
Academic	Administrative	0.38536	0.14949	0.165
Academic	Student	0.39949	0.08448	0.002
Administrative	Student	0.01413	0.13027	0.996

Table A.10: Tukey test between the groups (teachers, administrative staff, and students) and the RPS scale.

Group 1	Group 2	Media	Standard Error	P-value
Academic	Administrative	0.01278	0.151668	0.998
Academic	Student	0.29214	0.085712	0.044
Administrative	Student	0.30493	0.132167	0.235

Based on the previous analysis, there is a significant difference in the mean obtained (from 1 to 5) between the groups surveyed. Thus, teachers behave more conservatively on

the CBS scale in risky situations. Their mean is 3.067. On the RPS scale, the teachers obtained a mean of 3.780. Therefore, it is concluded that they have a better perception of risk when using the Internet or IT devices than the other groups surveyed.

Hypothesis H3: Exposure to more hours per day that users have when using the Internet affects the average of the scales RBS, CBS, EOS, RPS.

In contrast, H3 respondents were grouped into three ranges according to the time they use the Internet (i.e., 1 to 5 hours/day, 6 to 10 hours/day, and 11 or more hours/day). Table A.11 shows the ANOVA analysis, in which it can be identified if there is a significant difference in the case of the RBS scale with a value of P = 0.003. Therefore, there is a substantial difference between the participants' Internet use times. For this reason, H3 is rejected, and the alternative hypothesis is accepted, establishing that the surveyed users' Internet use affects the average of the proposed scales.

Table A.11: Results of the ANOVA of time of Internet use.

Scales						
RBS	Between groups	2.68	2	1.34	5.95	0.003
	Within groups	33.75	150	0.22	-	-
	Total	36.43	152	0.24	-	-
CBS	Between groups	0.12	2	0.59	0.24	0.78
	Within groups	36.20	150	0.24	-	-
	Total	36.32	152	0.24	-	-
EOS	Between groups	0.10	2	0.05	0.85	0.43
	Within groups	9.03	150	0.06	-	-
	Total	9.12	152	0.06	-	-
RPS	Between groups	0.66	2	0.33	1.37	0.26
	Within groups	35.95	150	0.24	-	-
	Total	36.61	152	0.24	-	-

To identify in which group this difference is generated, the Tukey test was applied between the groups that use the Internet in the three ranges (i.e., 1 to 5 hours/day, 6 to 10 hours/day, and 11 or more hours/day). The results obtained with the development of the Tukey test within the RBS scale reveal that the respondents in the 11 or more hours/day group are more exposed. In addition, they are more tolerant of risk situations than the

groups that use less time on the Internet, with a mean = 3.01268 on the RBS scale. See Table A.12.

Table A.12: Pearson correlation between the behavior scales.

Group 1	Group 2	Media	Std err	P-value
1-5 hours/day	6-10 hours/day	0.14	0.12	0.691
1-5 hours/day	11 or more hours/day	0.38	0.12	0.063
6-10 hours/day	11 or more hours/day	0.24	0.06	0.006

In conclusion, based on the analysis of the results obtained, it is shown that on the RBS scale, respondents in the group of 11 or more hours/day, with a mean = 3.012 on the RBS scale, are more exposed and are more permissive in risk situations, compared to groups that use the Internet for less time.

Hypothesis H4: There is a significant correlation between the averages of the scales RBS, CBS, EOS, and RPS.

In this study, the existing Pearson correlation between the scales was analyzed to test whether or not there is a correlation. The correlation is positive if the value obtained when comparing the scales is greater than zero and less than one or negative if the value obtained by comparing the scales is less than zero and greater than minus one. Table A.13 represents the data obtained by comparing the scales using Pearson's correlation.

Table A.13: Pearson correlation between the behavior scales.

Scales	RBS	CBS	EOS	RPS
RBS	1	0.17554	0.00705	0.23947
CBS	0.17554	1	-0.20972	0.381002
EOS	0.00705	-0.20972	1	-0.12996
RPS	0.23947	0.381002	-0.12996	1

Table A.13 shows the highest positive correlation when comparing the CBS and RPS scales, obtaining an r = 0.38. This means that if the CBS average increases, so will the RPS average. In other words, if conservative behavior increases in a user, they will have a better perception of risk when using the Internet or IT devices. On the other hand, the negative correlations obtained when developing the Pearson correlation occurred between

the CBS and EOS scales with r = -0.20 and between the EOS and RPS scales with r = -0.12. Hence, this indicates that if the average of CBS increases, the average of the EOS scale will decrease. In conclusion, if conservative behavior increases in a user, they will be less exposed to possible risks when using the Internet or IT devices. By doing the same analysis with the case of the EOS and RPS scales, if a user's exposure to offenses or risks increases, their perception of risk will decrease.

A.3. Annex Summary

In the first section, a framework was proposed to determine which people are more likely than others to be victims of a Phishing attack. This determination is achieved by evaluating each user characteristic as it fits the personality scale of the Five-Factor Model. Although the first section evaluates personality traits, these traits go hand in hand with people's behavior. Therefore, in the second section, people's behaviors and making them vulnerable to Social Engineering attacks were evaluated on a psychological scale. These results are useful in determining which people or groups of an organization are most likely to be victims of social engineering. This information is useful for monitoring or organizing training targeting these user groups.

Finally, based on the study in the first section of this chapter, it was determined that people with high Extraversion, Agreeableness, Openness, and low Conscientiousness are at a high risk of being victims of a social engineering attack. On the other hand, in the second section, it was determined that people who spend more time in front of a computer and are more permissive of risky behaviors are more vulnerable to these attacks.