People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
SAAD DAHLAB UNIVERSITY OF BLIDA 1
Faculty of Sciences
Department of Computer Science

DOCTORAL THESIS

Speciality : Network and security

Presented by :

REMMIDE Mohamed Abdelkarim

THEME

Hybrid approach to detect social engineering attacks

Publicly defended on 16/12/2024, before a jury composed of :

Prof.	Nadjia Benblidia	Professeur	USDB	Chair
Dr.	Fatima Boumahdi	MCA	USDB	Thesis supervisor
Prof.	Narhimene Boustia	Professeur	USDB	Thesis co-supervisor
Dr.	Massaouda Fareh	MCA	USDB	Examiner
Dr.	Ahmed Cherif Mazari	MCA	University of Medea	Examiner
Dr.	Fouaz Berrhail	MCA	University of Setif 1	Examiner

Acknowledgements

First and foremost, I express my profound gratitude to Almighty Allah for bestowing upon me the courage, strength, and perseverance necessary to undertake and complete this research endeavor. His guidance and blessings have been a constant source of support throughout this challenging journey.

I would like to convey my deepest appreciation to my supervisor, Dr. Fatima Boumahdi, and co-supervisor, Professor Narhimene Boustia, for their unwavering support, invaluable insights, and tireless guidance. Their expertise, constructive feedback, and commitment to academic excellence have been instrumental in shaping the direction and enhancing the quality of this thesis. Their mentorship has not only contributed to the successful completion of this work but has also significantly influenced my growth as a researcher.

I am profoundly grateful to the members of my thesis committee for their time, expertise, and thoughtful contributions. Their interdisciplinary perspectives, insightful critiques, and challenging questions have greatly enriched the depth and rigor of this research. Their commitment to academic excellence has pushed me to refine my ideas and strengthen the overall quality of this thesis.

My sincere gratitude extends to the faculty and staff of the Computer Sciences Department for fostering an intellectually stimulating and collaborative environment. The resources, facilities, and academic support provided have been crucial in facilitating this research.

I would also like to acknowledge the invaluable support of my colleagues and fellow researchers in the cybersecurity lab. Our discussions, collaborations, and mutual support have been a source of inspiration and have contributed significantly to the ideas presented in this thesis.

Finally, I am deeply indebted to my family for their unconditional love, understanding, and encouragement throughout this challenging journey. Their unwavering belief in my abilities has been a constant source of motivation.

This thesis stands as a testament to the collective support, guidance, and encouragement of all those mentioned above and many others who have contributed in various ways. To each of you, I express my heartfelt gratitude.

ملخص

أدت التحولات الرقمية السريعة في المجتمع الحديث إلى تحقيق مستويات غير مسبوقة من الاتصال والابتكار، بينما أدت أيضاً إلى ظهور تحديات معقدة في مجال الأمن السيبراني. ومن بين هذه التحديات، أصبحت هجمات الهندسة الاجتماعية، وخاصة التصيد الاحتيالي (Phishing)، تهديداً واسع الانتشار، حيث تستغل نقاط الضعف البشرية لتجاوز إجراءات الأمن التقليدية. تقدم هذه الأطروحة بحثاً متعمقاً حول التقنيات المتقدمة لاكتشاف وتخفيف هجمات التصيد الاحتيالي عبر عدة نواقل، بما في ذلك البريد الإلكتروني والعناوين الإلكترونية (URLs) والرسائل النصية القصيرة (SMS).

من خلال الاعتماد على منهجيات متطورة في التعلم الآلي والتعلم العميق، تساهم هذه البحث بعدة إسهامات رئيسية في مجال الأمن السيبراني. أولاً، قمنا بتحسين اكتشاف هجمات الهندسة الاجتماعية من خلال تطبيق تقنيات زيادة العينات (Oversampling) مع آلات ناقلات الدعم (Support Vector Machines - SVM)، مع التحقق من صحتها باستخدام اختبارات تحليل التباين الإحصائي (Analysis of Variance - ANOVA). يُعد تحليل التباين (ANOVA) أسلوباً إحصائياً يُستخدم لمقارنة أداء نماذج متعددة من خلال تحليل التباين بين نتائجها، مما يضمن قوة وموثوقية الحلول المقترحة.

ثانياً، نقدم تطبيقاً مبتكراً لشبكات الالتفاف الزمنية (Temporal Convolutional Networks TCN) لاكتشاف عناوين URL التصيدية، حيث تُظهر أداءً متفوقاً في تحليل الأنماط التسلسلية لأحرف العناوين الإلكترونية. ثالثاً، نقترح نهجاً هجيناً يدمج بين التعلم العميق والاستدلال القائم على الحالات (Case-Based Reasoning - CBR)، مما يتيح استراتيجيات تكيفية لاكتشاف التصيد الاحتيالي بناءً على أنماط الهجمات السابقة.

لمواجهة التطور المستمر في تهديدات التصيد الاحتيالي، يستكشف هذا العمل استخدام الشبكات العصبية السيامية (Siamese Neural Networks) للتحقق من صحة المؤلف في تصنيف البريد الإلكتروني، كما يتم تنفيذ إطار عمل للتعلم العميق الموحد مع الحفاظ على الخصوصية (Privacy-Preserving Federated Deep Learning) لاكتشاف هجمات التصيد عبر الرسائل النصية القصيرة (Smishing).

يوفر نهجنا الشامل فهماً أعمق للنظريات الكامنة وراء اكتشاف التصيد الاحتيالي، كما يقدم حلولاً عملية وقابلة للتطبيق لتعزيز إجراءات الأمن السيبراني. من خلال معالجة الطبيعة متعددة الأوجه لهجمات التصيد والاستفادة من التقنيات الحاسوبية المتقدمة، تساهم هذه الأبحاث في تطوير آليات دفاعية أكثر قوة وقدرة على التكيف ضد التهديدات السيبرانية المتطورة باستمرار في نظامنا الرقمي المترابط بشكل متزايد.

الكلمات المفتاحية: الهندسة الاجتماعية؛ هجمات التصيد الاحتيالي؛ اكتشاف رسائل التصيد الاحتيالي عبر البريد الإلكتروني؛ التعلم العميق؛ تمثيل المعرفة؛ الاستدلال القائم على الحالات؛ التعلم الفيدرالي.

Abstract

The rapid digital transformation of modern society has ushered in unprecedented connectivity and innovation, while simultaneously introducing complex cybersecurity challenges. Among these, social engineering attacks, particularly phishing, have emerged as a pervasive threat, exploiting human vulnerabilities to bypass traditional security measures. This thesis presents a comprehensive investigation into advanced techniques for detecting and mitigating phishing attacks across multiple vectors, including emails, URLs, and SMS messages.

Leveraging state-of-the-art machine learning and deep learning methodologies, this research makes several key contributions to the field of cybersecurity. First, we advance the detection of social engineering attacks through the application of oversampling techniques with Support Vector Machines (SVM), validated using Analysis of Variance (ANOVA) statistical tests. ANOVA is a statistical method used to compare the performance of multiple models by analyzing the variance between their results, ensuring the robustness and reliability of our proposed solutions.

Second, we introduce a novel application of Temporal Convolutional Networks (TCNs) for phishing URL detection, demonstrating superior performance in analyzing sequential patterns of URL characters. Third, we propose a hybrid approach that integrates deep learning with case-based reasoning (CBR), enabling adaptive phishing email detection strategies informed by historical attack patterns.

To address the evolving landscape of phishing threats, this work explores Siamese neural networks for authorship verification in email classification and implements a privacy-preserving federated deep learning framework for detecting SMS-based phishing (smishing) attacks.

Our comprehensive approach not only advances the theoretical understanding of phishing detection but also provides practical, implementable solutions to enhance cybersecurity measures. By addressing the multifaceted nature of phishing attacks and leveraging advanced computational techniques, this research contributes to the development of more robust and adaptive defense mechanisms against evolving cyber threats in our increasingly interconnected digital ecosystem.

Keywords: Social engineering; Phishing attacks; Phishing email detection; Deep learning; knowledge representation; Case-based Reasoning; federated learning.

Résumé

La transformation numérique rapide de la société moderne a engendré une connectivité et une innovation sans précédent, tout en introduisant des défis complexes en matière de cybersécurité. Parmi ces défis, les attaques d'ingénierie sociale, en particulier le phishing, sont devenues une menace omniprésente, exploitant les vulnérabilités humaines pour contourner les mesures de sécurité traditionnelles. Cette thèse présente une investigation approfondie des techniques avancées pour détecter et atténuer les attaques de phishing sur plusieurs vecteurs, notamment les e-mails, les URL et les SMS.

En s'appuyant sur des méthodologies de pointe en apprentissage automatique et apprentissage profond, cette recherche apporte plusieurs contributions majeures au domaine de la cybersécurité. Premièrement, nous améliorons la détection des attaques d'ingénierie sociale grâce à l'application de techniques de suréchantillonnage avec les Machines à Vecteurs de Support (SVM), validées par des tests statistiques d'Analyse de Variance (ANOVA). L'ANOVA est une méthode statistique utilisée pour comparer les performances de plusieurs modèles en analysant la variance entre leurs résultats, garantissant ainsi la robustesse et la fiabilité de nos solutions proposées.

Deuxièmement, nous introduisons une application novatrice des Réseaux Convolutionnels Temporels (TCN) pour la détection des URL de phishing, démontrant une performance supérieure dans l'analyse des motifs séquentiels des caractères d'URL. Troisièmement, nous proposons une approche hybride qui intègre l'apprentissage profond avec le raisonnement à base de cas (CBR), permettant des stratégies de détection de phishing adaptatives basées sur des modèles d'attaques historiques.

Pour répondre à l'évolution des menaces de phishing, ce travail explore les réseaux de neurones à architecture Siamese pour la vérification de la paternité dans la classification des e-mails et met en œuvre un cadre d'apprentissage profond fédéré préservant la confidentialité pour détecter les attaques de phishing par SMS (smishing).

Notre approche globale permet non seulement d'approfondir la compréhension théorique de la détection du phishing, mais offre également des solutions pratiques et applicables pour renforcer les mesures de cybersécurité. En abordant la nature multifacette des attaques de phishing et en exploitant des techniques informatiques avancées, cette recherche contribue au développement de mécanismes de défense plus robustes et adaptatifs contre les cybermenaces en constante évolution dans notre écosystème numérique de plus en plus interconnecté.

Mots Clée : Ingénierie sociale ; Attaques de phishing ; Détection des e-mails de phishing ; Apprentissage profond ; Représentation des connaissances ; Raisonnement à base de cas ; Apprentissage fédéré.

Contents

List of Figures

List of Tables

List of Abbreviations

Acronyms / Abbreviations

AI	Artificial intelligence
ANOVA	Analysis of Variance
APWG	Anti-Phishing Working Group
ASSET	Anti-Social Engineering Tool
BERT	Bidirectional Encoder Representations from Transformers
BgC	Biosynthetic Gene Clusters
Bi-LSTM	Bidirectional LSTM
BTC	Bitcoin
CBR	Case-Based Reasoning
CD	Compact disc
CNN	Convolutional Neural Network
DBIR	Data Breach Investigation Report
DL-CBR	Deep Learning-augmented Case-Based Reasoning
ETC	Extended Transformer Construction
GBDT	Gradient-boosted decision trees
GCN	Graph Convolutional Network
GNB	Gaussian Naive Bayes
GRU	Gated Recurrent Unit

KNN	K-Nearest Neighbor
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MNB	Multinomial Naive Bayes
NB	Naive Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
PCA	Principal Component Analysis
PRISM	Personal grouping method for Reducing InterSubject variability
PSD	Path-Similarity Distance
RBF	Radial Basis Functio
RCNN	Region-based Convolutional Neural Network
ResNets	Residual Networks
RF	Random Forest
RNN	Recurrent Neural Network
SE	social engineering
SEADM	Social Engineering Attack Detection Mode
SGB	Stochastic Gradient Descent
Smishing	SMS-based phishing
SMOTE-ENN	Synthetic Minority Over-sampling Technique with Edited Nearest Neighbors
SVC	Support Vector Classifier
SVM	Support Vector Machine
TCN	Temporal Convolutional Network

TF	Term Frequencies
TF-IDF	Term frequency-inverse document frequency
URL	Uniform Resource Locator
USB	Universal Serial Bus
Vishing	voice-based phishing
VPN	Virtual Private Network
XGB	EXtreme Gradient Boosting

General Introduction

The rapid advancement of digital technologies has profoundly transformed modern society, permeating every facet of daily life. According to the 2022 Global Digital Report, an unprecedented 63.5% of the global population—over 5 billion individuals—regularly engage with the internet for a diverse array of activities, including communication, e-commerce, and digital education [Digital GLOBAL OVERVIEW REPORT, 2022]. This statistic not only underscores the pervasive nature of digital technology but also highlights the critical role the internet plays in facilitating and mediating human experiences in the 21st century.

While this digital transformation offers unparalleled opportunities for connectivity and innovation, it has concurrently ushered in a complex landscape of cybersecurity challenges. The escalation of these challenges is evident in recent reports. Deep Instinct observed a staggering 653% increase in malicious cybersecurity activity in July 2020 compared to the previous year [Cyber Threat, 2020]. Furthermore, the U.S. Healthcare Cybersecurity Market report revealed that over 90% of healthcare organizations experienced at least one cybersecurity breach in the past three years [1]. These findings illustrate the growing vulnerability of critical sectors to cyber threats, emphasizing the urgent need for robust and adaptive security measures.

Problem statement

Despite significant advancements in technological security measures, the human factor remains one of the most critical vulnerabilities in cybersecurity ecosystems. According to Verizon's 2021 Data Breach Investigation Report (DBIR), 85% of data breaches are attributed to human-related factors [DBIR Report, 2021]. This vulnerability is systematically exploited through social engineering attacks, which manipulate human psychology to bypass traditional security protocols. These attacks prey on cognitive biases, lack of awareness, and the inherent trust individuals place in digital communications, making them both pervasive and difficult to mitigate.

[1]https://www.gminsights.com/industry-analysis/healthcare-cybersecurity-market

Among social engineering tactics, phishing has emerged as a particularly pernicious and widespread threat. The Anti-Phishing Working Group (APWG) reported an unprecedented surge in phishing attacks, with over 1 million incidents recorded in the first quarter of 2022 alone [APWG, 2021a]. The financial sector remains the primary target, accounting for 23.2% of all phishing attacks [APWG, 2021a]. The economic impact of these attacks is staggering, with losses estimated at billions of dollars annually. Beyond financial damage, phishing attacks lead to data breaches, identity theft, compromised critical infrastructure, and a erosion of trust in digital systems.

The evolving nature of phishing attacks further complicates the challenge. While email remains the most common vector, attackers have diversified their methods to include: Smishing (SMS-based phishing),Vishing (voice-based phishing), andExploitation of social media platforms [Gupta et al., 2018].

This diversification underscores the need for a multifaceted and adaptive approach to detection and prevention. Traditional security measures, such as blacklists and rule-based systems, are often ineffective against these evolving tactics, as they rely on static patterns and cannot adapt to zero-day attacks or novel social engineering techniques.

The human factor exacerbates these challenges. Cognitive biases, such as the tendency to trust authoritative-looking communications, and a general lack of cybersecurity awareness make individuals easy targets for sophisticated social engineering attacks. This not only compromises individual security but also undermines trust in digital ecosystems, with far-reaching consequences for businesses, governments, and society as a whole.

In light of these challenges, there is an urgent need for innovative solutions that address the human factor, adapt to the evolving tactics of attackers, and provide robust detection and prevention mechanisms across multiple attack vectors. This thesis seeks to address these gaps by leveraging machine learning, deep learning, and hybrid approaches to develop advanced techniques for combating phishing.

Research methodology

To address these evolving threats, this thesis employs cutting-edge machine learning and deep learning methodologies. The research draws upon and extends existing literature, which can be broadly categorized into three solution types [Fang et al., 2019]: [Rao and Pais, 2017], machine learning [Alam et al., 2020], and deep learning [Remmide et al., 2022a]. While blacklists have shown promise, they are limited by their reliance on human maintenance and inability to detect zero-day phishing attempts. Machine learning techniques have demonstrated effectiveness but often require expert-driven feature engineering. Deep learning approaches have emerged as particularly promising, offering effective results without the need for extensive feature engineering.

This thesis aims to develop and evaluate advanced techniques for detecting phishing attacks across multiple vectors, including email, URLs, and SMS. The research leverages a combination of novel algorithms, hybrid models, and privacy-preserving frameworks to address the limitations of existing methods. The key contributions of this work are as follows:

- **Advancing Automated Social Engineering Detection with Oversampling-Based Machine Learning:** This study utilizes oversampling techniques with Support Vector Machines (SVM) for the detection of social engineering attacks. Additionally, Analysis of Variance (ANOVA) is employed to assess the effectiveness of the model [Remmide et al., 2024b].
- **Phishing URL Detection Using Temporal Convolutional Networks (TCN):** This approach leverages Temporal Convolutional Networks to analyze the sequence of characters in URLs, identifying patterns that are characteristic of phishing attempts [Remmide et al., 2022b].
- **Towards a Hybrid Approach Combining Deep Learning and Case-Based Reasoning for Phishing Email Detection:** Integrating deep learning models with case-based reasoning to enhance phishing email detection by leveraging past instances of phishing attacks to inform current detection strategies [Remmide et al., 2024c].
- **Privacy-Preserving Smishing Detection with Federated Deep Learning:** A federated deep learning approach is used to detect smishing attacks while preserving user privacy, enabling collaborative learning across multiple devices without exposing sensitive data [Remmide et al., 2024d].
- **Authorship Verification for Phishing Email Detection Using Siamese Neural Networks:** Employing Siamese neural networks to verify email authorship, this approach detects phishing attempts by identifying inconsistencies in writing style when compared to legitimate communications [Remmide et al., 2024a].

Through these contributions, this thesis aims to advance the field of cybersecurity by addressing critical gaps in phishing detection. The proposed solutions not only improve detection accuracy but also enhance adaptability to emerging threats, ensuring robust defense mechanisms in an increasingly interconnected digital world. By leveraging the strengths of machine learning, deep learning, and hybrid approaches, this research provides a comprehensive framework for combating the evolving threat of phishing attacks.

Thesis organization

This thesis is structured into two main parts: the ***State of the Art***, which offers an overview of the thesis context and a review of related works, and the ***Contribution*** part, which presents the novel approaches developed in this research to address social engineering attacks, particularly phishing.

The State of the Art part lays the theoretical foundation for the thesis, exploring the landscape of social engineering attacks and existing detection methods. It comprises the following chapters::

- Chapter 1: This chapter explores the concept of social engineering, its psychological underpinnings, and the various types of attacks, including phishing, smishing, and vishing. It also discusses the evolving tactics used by attackers and their impact on individuals and organizations.
- Chapter 2: This chapter provides a detailed examination of email phishing, including the anatomy of phishing attacks, common techniques used by attackers, and existing detection methods. It highlights the limitations of current approaches and sets the stage for the proposed solutions.
- Chapter 3: This chapter introduces the theoretical frameworks underpinning the proposed solutions, with a focus on case-based reasoning (CBR) and deep learning.

Part II presents the original contributions of the thesis, each addressing specific aspects of phishing detection. It includes the following chapters:

- Chapter 4: This chapter explores the use of oversampling techniques combined with SVM to detect social engineering attacks. The approach addresses class imbalance in datasets, enhancing the accuracy and reliability of detection. The effectiveness of the model is evaluated using Analysis of Variance (ANOVA) to rank its performance.
- Chapter 5: This chapter focuses on detecting phishing URLs using TCNs, a deep learning architecture well-suited for sequence modeling. The approach analyzes the sequential structure of URLs to identify patterns characteristic of phishing attempts, contributing to enhanced security measures.

- Chapter 6: CBR is combined with deep learning to solve the problem of phishing email detection. First, deep learning generates the case representation using the N-pair loss function, which comes from the field of deep metric learning. With the goal of grouping similar emails into the same cluster. The model consists of a pre-trained GloVe embedding and a TCN+Bi-LSTM network with an attention mechanism followed by a CBR classifier.

- Chapter 7: This chapter addresses the detection of smishing using federated learning. The approach enables collaborative learning across multiple devices while preserving user privacy, ensuring that sensitive data is not exposed during the detection process.

- Chapter 8: This chapter introduces a novel method for detecting phishing emails by verifying authorship using Siamese Neural Networks. The model employs LSTM layers and a Manhattan distance layer to measure the similarity between emails, capturing unique writing styles and language patterns to identify inconsistencies indicative of phishing attempts.

- The final chapter summarizes the key findings of the thesis, discusses their implications for the field of cybersecurity, and outlines potential directions for future research.

Part I

State of the art

Chapter 1

Social engineering

1.1 Introduction

Advancements in technology are rapidly shaping modern society, exerting significant influence on daily life. A 2022 Global Digital Report revealed that over 5 billion people, or 63.5% of the global population, regularly use the internet for activities such as communication, online shopping, and digital education [Digital GLOBAL OVERVIEW REPORT, 2022]. This highlights our growing reliance on technology, especially the internet.

While greater connectivity has created new opportunities, it has also introduced serious security challenges. The 2022 Verizon Data Breach Investigations Report noted a 13% rise in ransomware attacks, exploiting the expanded use of the internet [DBIR Report, 2022].

This increasing dependence on technology, coupled with rising cyber threats, sets the stage for social engineering attacks. Social engineering exploits human psychology to access sensitive information or systems, becoming a critical concern in the digital age. As our lives become more intertwined with technology, the potential impact of these attacks grows exponentially.

This chapter explores the concept of social engineering, its attack vectors, and the psychological principles that make it effective. It also examines various types of attacks and reviews the state of the art in detection and prevention techniques.

1.2 Understanding social engineering

Social engineering is a sophisticated manipulation technique that exploits human psychology to gain unauthorized access to confidential information or systems [Kamruzzaman et al., 2023]. It operates by deceiving individuals into bypassing standard security procedures, often through carefully crafted scenarios that leverage trust, urgency, or fear [Kamruzzaman et al., 2023]. This method has evolved from a traditional psychological concept into a major cyber-attack vector, primarily due to human error and a lack of awareness about cyber threats among employees and users.

The effectiveness of social engineering lies in its exploitation of human nature. Attackers skillfully use psychological triggers to manipulate their targets. They can occur online, in-person, or through various interactions. They may induce emotions like fear, excitement, or sympathy; create a sense of urgency to force quick, poorly considered decisions; impersonate figures of authority or influence; or exploit existing relationships and build false ones to gain trust. By leveraging these psychological vulnerabilities, social engineers can bypass sophisticated technical security measures, making their attacks particularly potent and challenging to defend against. This reliance on human psychology rather than technical exploits sets social engineering apart from many other forms of cyberattacks and underscores the importance of human-focused security measures.

These techniques capitalize on the human tendency to trust others and make rapid decisions under pressure, making social engineering attacks particularly potent and challenging to defend against [Salahdine and Kaabouch, 2019]. The impact of social engineering extends beyond individual victims. Organizations face significant risks, including data breaches, financial losses, and reputational damage. As our reliance on digital systems grows, so does the potential impact of these attacks. This underscores the critical need for comprehensive cybersecurity strategies that address not only technological vulnerabilities but also human factors.

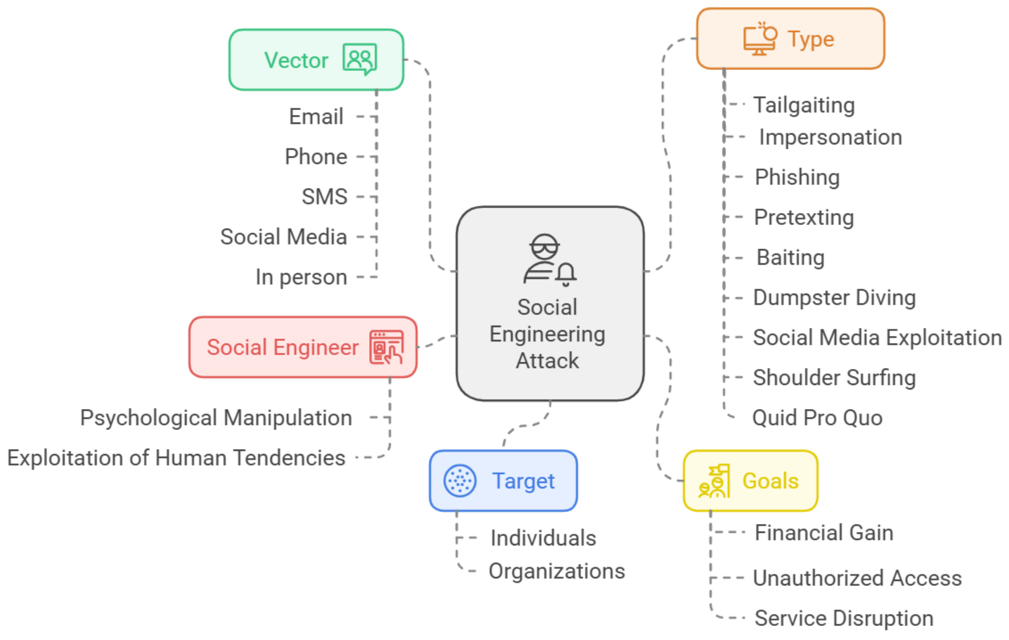


Figure 1.1: Overview of Social Engineering Attack Structure.

Figure 1.1 presents a comprehensive model of a social engineering attack. The social engineer, whether an individual or group, exploits various communication mediums such as email, SMS, telephone, or webpages to target individuals or organizations. The attack's goal may include financial gain, unauthorized access, or service disruption. These attacks are driven by principles of compliance, leveraging psychological tactics like reciprocity, authority, and social validation to manipulate targets. Specific techniques, including phishing, pretexting, and baiting, are used depending on the mode of communication and the desired outcome. This holistic view emphasizes the multifaceted nature of social engineering and the sophistication of its methods.

1.3 Social Engineering Attack Vectors

At its core, social engineering involves manipulating people into performing actions or divulging sensitive information, such as passwords, bank details, or access codes. These attacks can be carried out through various channels, known as attack vectors. Common attack vectors, as shown in Figure 1.2 include :

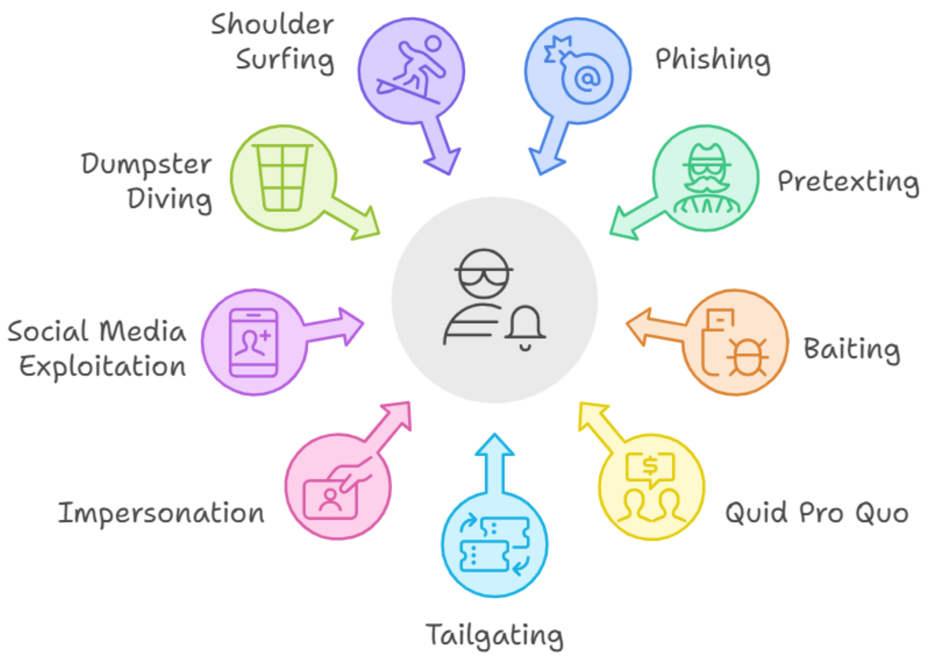


Figure 1.2: Social Engineering Attack Vectors

- **Email-based Attacks (Phishing) :**
  Email-based attacks, commonly known as phishing, involve the attacker sending fraudulent emails that appear to be from legitimate sources, such as organizations or individuals, in an attempt to trick the recipient into revealing sensitive information

or performing a desired action, such as clicking on a malicious link or downloading an infected attachment.

- **Phone-based Attacks (Vishing)**
  Vishing, or voice phishing, involves the attacker using the telephone to social engineer their target. The attacker may impersonate a customer service representative, IT support, or other authority figure to convince the target to disclose sensitive information or perform a specific action, such as transferring funds or providing login credentials.

- **SMS-based Attacks (Smishing) :**
  Smishing attacks use text messages (SMS) to lure the target into disclosing sensitive information or clicking on a malicious link. These attacks often mimic messages from trusted organizations or individuals to exploit the target's trust and sense of urgency.

- **Social Media Exploitation :**
  Attackers may leverage social media platforms to gather information about their targets, such as their interests, relationships, and online behavior. This information can then be used to craft more personalized and convincing social engineering attempts, such as spear-phishing emails or targeted scams.

- **In-person Attacks :**
  In-person social engineering attacks involve the attacker physically interacting with the target to manipulate them into divulging sensitive information or performing a desired action. Examples include tailgating (following an authorized person through a secure door) or dumpster diving (searching for discarded documents containing sensitive information).

This diverse range of attack vectors demonstrates the versatility and adaptability of social engineering techniques, making them a formidable threat in both digital and physical environments.

1.4 Attack cycle

Social engineering attacks, while differing in specific techniques, generally follow a consistent pattern with four distinct phases. According to [Mitnick and Simon, 2003], these phases include: (1) collecting information about the target, (2) building a relationship with the target, (3) using the acquired information to carry out the attack, and (4) making a clean exit without leaving any trace. Figure 1.3 depicts the various stages involved in a typical social engineering attack.

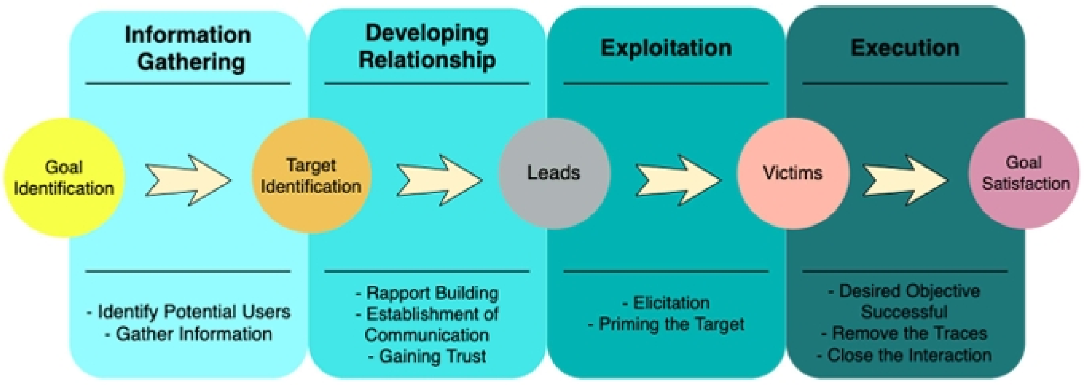


Figure 1.3: Social Engineering Attack cycle [Mashtalyar et al., 2021]

- **Investigation:** This initial phase involves gathering information about potential victims. Attackers research their targets to identify weaknesses and determine the best methods for manipulation. This can include studying social media profiles, organizational structures, and employee behaviors to find vulnerabilities that can be exploited later.

- **Hook:** In this phase, the attacker establishes contact with the victim, often using tactics designed to build trust. This could involve impersonating a colleague or authority figure to create a sense of familiarity and credibility. The goal is to engage the victim in a way that makes them more susceptible to manipulation.

- **Play:** Here, the attacker executes the actual manipulation or deception. This could involve phishing emails, phone calls, or other forms of communication where the victim is tricked into providing sensitive information or performing actions that compromise security. The attacker capitalizes on the trust established in the previous phase to achieve their objectives.

- **Exit:** After successfully obtaining the desired information or access, the attacker disengages from the victim. This phase often includes erasing traces of their activities to avoid detection and ensure they can exploit the information gained without being caught

1.5 Psychological manipulation and exploitation

Psychological manipulation and exploitation in social engineering attacks involve the use of psychological tactics to influence and control individuals, often leading them to take actions that compromise their security. These attacks exploit human emotions, cognitive biases, and social norms to achieve malicious goals. They also employ persuasion techniques to distract individuals from thinking critically and analytically by overwhelming them with disinformation within the normal flow of communication [Ross et al., 2021].

Renowned psychologist Robert Cialdini proposed a well-known taxonomy of six core principles of persuasion [Resnik, 1986]: Authority, Scarcity, Liking/Similarity, Reciprocation, Social Proof, and Commitment/Consistency. Later, he expanded his work by introducing a seventh principle, Unity, which emphasizes shared identity and group belonging [Cialdini Robert, 2021].

- **Reciprocity:**
  This principle is based on the idea that humans feel obligated to return favors or acts of kindness. When someone does something for us, we feel compelled to reciprocate. In persuasion, this might involve offering something of value first (like free samples or helpful information) before making a request.

- **Commitment and Consistency:**
  Once people take a stand or go on record in favor of a position, they prefer to stick to it. This desire to be (and appear) consistent with our words, beliefs, attitudes, and deeds is a highly potent weapon of social influence. Marketers often exploit this by getting people to make small commitments before larger ones.

- **Authority:**
  People tend to obey authority figures, even sometimes when asked to perform objectionable acts. This principle explains why people are more likely to be persuaded by individuals perceived as credible, knowledgeable, or authoritative in a particular field. Titles, clothing (like uniforms), and trappings of authority can increase perceived authority.

- **Liking:**
  We're more likely to be influenced by people we like. Cialdini identified several factors that contribute to liking:Physical attractiveness, Similarity (to ourselves), Compliments, Contact and cooperation, Conditioning and association.

- **Social Proof:**
  This principle states that people tend to look to the actions of others to determine

their own, especially in uncertain situations. We assume that if lots of other people are doing something, it must be the correct thing to do. This is why testimonials, popularity claims, and "bestseller" labels are so effective.

- **Scarcity:**
  The principle of scarcity states that people want more of what they can have less of. When something is rare or becoming rare, it's perceived as more valuable. This principle is often used in marketing with "limited time offers" or "while supplies last" messaging.

- **Unity:**
  This principle, added later by Cialdini, suggests that people are more likely to say yes to those who they see as part of their in-group. It goes beyond simple similarities to shared identities. This could be based on ethnicity, nationality, family relations, or shared experiences.

1.6 Types of social engineering attacks

Social engineering tactics used by attackers encompass various methods to deceive individuals into divulging sensitive information. These tactics include phishing, quid pro quo, pretexting, baiting, and tailgating, which exploit human psychology and behavior to manipulate victims [Kamruzzaman et al., 2023]. Figure 1.4 summarizes these types of social engineering attacks.

- **Phishing :**
  Phishing is the most common type of social engineering attack, where attackers send fraudulent emails impersonating legitimate organizations or individuals. These emails often contain malicious links or attachments designed to steal sensitive information or install malware on the victim's device. Spear phishing and whaling are targeted variations of phishing that focus on specific individuals or high-level executives, respectively [Kamruzzaman et al., 2023].

- **Pretexting :**
  Pretexting involves creating a plausible pretext or scenario to convince the target to provide the desired information or perform a specific action. Attackers may pose as a customer, IT support, or other authority figure to gain the target's trust [Applegate, 2009].

- **Baiting :**
  Baiting involves leaving physical media, such as USB drives or CDs , containing malware in a public place, hoping that the target will find and use the device, infecting their system [Kamruzzaman et al., 2023].

- **Quid pro quo :**
  Quid pro quo attacks involve the attacker offering the target a benefit, service, or item of value in exchange for the target providing the attacker with access, information, or some other desired outcome [Krombholz et al., 2015]. The attacker's goal is to exploit the target's desire for the offered benefit to manipulate them into compromising security measures or disclosing sensitive data.

- **Tailgating :**
  Tailgating, also known as "piggybacking," occurs when an attacker follows an authorized person through a secure door or gate, gaining unauthorized access to a restricted area. The attacker may also distract the victim or claim to have forgotten their access card to gain unauthorized entry.

- **Impersonation :**
  Impersonation involves an attacker pretending to be a legitimate person, such as a co-worker, customer, or authority figure, to gain the target's trust and obtain sensitive information [Kamruzzaman et al., 2023].

- **Social Media Exploitation :**
  Attackers may use information publicly available on social media platforms to gather intelligence about the target, their relationships, and their interests, which can then be used to craft more convincing social engineering attempts [Edwards et al., 2017].

- **Dumpster Diving :**
  Dumpster diving is the practice of searching through an organization's trash or recycling for sensitive documents, printouts, or other information that can be used to launch social engineering attacks [Applegate, 2009].

- **Shoulder Surfing :**
  Shoulder surfing involves an attacker physically observing the target's activities, such as watching them enter passwords or other sensitive information, to gather information that can be used to gain unauthorized access [Applegate, 2009].

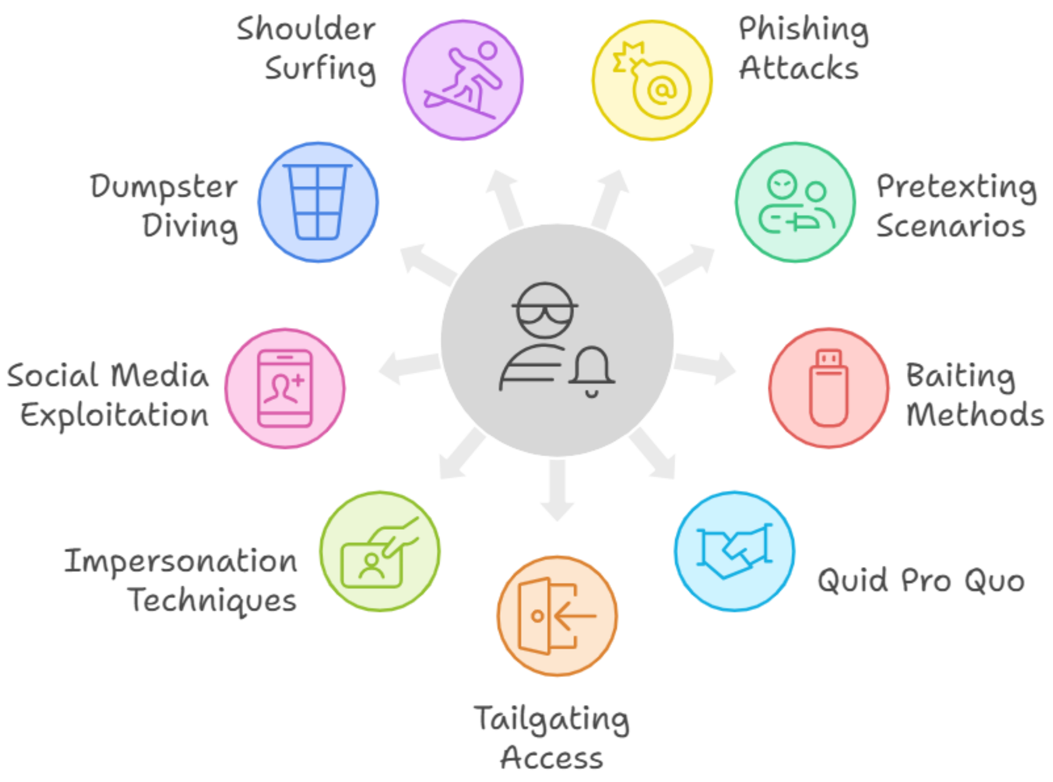


Figure 1.4: Social engineering attack methods

1.7 Detecting of social engineering

The threat of social engineering (SE) to information security is significant, as it targets individuals rather than technical systems. In today's world, SE attacks represent one of the most prevalent and costly threats to organizations.

Researchers have previously sought to develop methods for the automatic detection of SE attacks. A comprehensive overview of the state-of-the-art social engineering attack recognition systems can be found in Tsinganos et al. [Tsinganos et al., 2018]. They propose a novel approach that incorporates personality recognition, influence recognition, deception recognition, speech act analysis, and chat history.

Sawa et al. [Sawa et al., 2016] presented a proposal utilizing natural language processing to identify requests seeking private data, as well as instructions or commands that violate security policies. To identify potential social engineering attacks, the researchers manually generated lists of verb-noun combinations considered sensitive.

Building upon the investigation by Sawa et al. [Sawa et al., 2016], Peng et al. [Peng et al., 2018] identified four principal attack vectors relevant to social engineering: urgency of dialogue, negative or threatening commands or questions, automation cues (such as generic greetings), and URL safety checks. To create phishing blacklists, Peng et al. trained a Naive Bayes classifier using a large collection of recognized phishing emails, rather than relying on manual methods.

Understanding the linguistic strategies employed in social engineering attacks can aid in the development of detection methods. Derakhshan et al. [Derakhshan et al., 2021] found that social engineering attacks frequently involve manipulative speech acts. Consequently, they developed the Anti-Social Engineering Tool (ASSET), which identifies and understands the semantic dimensions involved in conversations.

Mouton et al. [Mouton et al., 2015] proposed an enhanced iteration of the Social Engineering Attack Detection Model (SEADM) [Mouton et al., 2014], which was subsequently validated through the use of published generalized social engineering examples. In another revision of the SEADM, Mouton et al. [Mouton et al., 2018] formalized social engineering attacks using a finite state machine.

Expanding on prior research, Bhatia et al. [Bhatia et al., 2020] proposed an extensible lexicon methodology based on lexical conceptual structure to improve the detection of social engineering attacks. Their framework aimed to encompass the extensive linguistic knowledge required for detecting and preventing modern manipulation techniques.

Lansley et al. [Merton Lansley and Polatidis, 2020] proposed using artificial neural networks, specifically multilayer perceptron classifiers (MLP), along with ensemble learning techniques, to identify instances of social engineering attacks that could occur both online and offline. Their method achieved a high accuracy rate of 92.6%.

In [Lopez and Camargo, 2022], the authors utilized Natural Language Processing (NLP) to extract features from dialog text, such as URL counts, spell checks, blacklist counts, and others. These features were used to train machine learning algorithms (Neural Network, Random Forest, and SVM) for classifying social engineering attacks. The paper reports that these classification algorithms achieved an accuracy exceeding 80%.

The authors in [Ozen et al., 2024] presented a comprehensive framework focusing on the visual traits of SE attack pages, which can differ from legitimate sites. The framework combines a custom security crawler, named SECrawler, designed to scout the web for examples of in-the-wild social engineering attacks. SENet, a deep learning-based image classifier, analyzes the visual traits of web pages associated with social engineering attacks, trained on data gathered by SECrawler. They also proposed SEGuard, a proof-of-concept browser extension that integrates SENet into web browsers. Through extensive evaluation, SENet reported a detection rate of up to 99.6%, with only a 1% false positive rate.

In [Tsinganos et al., 2022a], the authors introduced CSE-PersistenceBERT, a natural language processing model for paraphrase detection, focusing on identifying persistence as a key behavior of social engineering attackers in chat-based dialogues. For this purpose, they developed the specialized CSE-Persistence corpus.

Building on their earlier work, the same authors, in [Tsinganos et al., 2023], proposed a set of dialogue acts specific to chat-based social engineering (CSE) attacks, termed SG-CSE DAs. These acts are designed to reveal the attacker's intent and the type of information being targeted in conversations. In addition, they introduced a new annotated dataset, the SG-CSE Corpus, and developed SG-CSE BERT, a BERT-based model for zero-shot dialogue-state tracking in the context of CSE attacks.

Further expanding their research, the authors in [Tsinganos et al., 2022b] trained a convolutional neural network (CNN) on the CSE Corpus, a dataset annotated to recognize Cialdini's persuasion principles. The resulting classifier, named CSE-PUC, predicts whether a sentence carries a persuasive intent by generating a probability distribution over sentence classes, providing insight into how persuasion is used during social engineering attacks.

In [Lee et al., 2020], the authors present a context-aware approach to improve the detection of sophisticated phishing attempts. By fine-tuning a pre-trained BERT model to capture the syntactic and semantic nuances of natural language, their proposed model achieved an accuracy of 87% and demonstrated resilience against adversarial attacks, where attackers attempt to evade detection by replacing keywords with synonyms.

Similarly, [Lan, 2021] introduces a social engineering detection model based on a deep neural network, designed to identify deception and phishing attempts through text analysis. In the first stage, chat history is processed using natural language techniques, with context and semantics captured by a bi-LSTM model. The model further integrates user and chat content characteristics as features for classification, utilizing a ResNet architecture for improved accuracy.

In [Dalton et al., 2020], the authors propose a system that protects against CSE attacks by deploying a pipeline of NLP components, including Named Entity Recognition (NER), dialogue management, stylometry, and framing questions. This system uses an active defense strategy to recognize the social engineer's intentions, aiming to waste their time and resources.

Similarly, [Saleilles and Aïmeur, 2021] describes a different approach, introducing a chatbot designed to educate users and raise awareness of social engineering attacks. The chatbot first assesses users' knowledge of cybersecurity concepts through a quiz, then recommends specific training paths to address knowledge gaps. Throughout the training, the chatbot simulates malicious questioning techniques to extract sensitive information, enhancing users' awareness of potential threats.

1.8 Synthesis

Social engineering (SE) is a critical threat to information security, exploiting human vulnerabilities rather than technical systems. With SE attacks becoming one of the most prevalent and costly threats to organizations, researchers have developed various methods to detect and mitigate them. Early research focused primarily on rule-based methods. Sawa et al. [Sawa et al., 2016] used NLP techniques to manually identify sensitive verb-noun combinations, a method that struggled to adapt to evolving attacks and required significant manual effort.

As SE tactics became more sophisticated, researchers turned to machine learning approaches. Peng et al. [Peng et al., 2018] advanced the field by employing a Naive Bayes classifier to detect phishing through four attack vectors, such as urgency and URL safety checks. While this improved automation, its reliance on predefined training datasets posed limitations in detecting new or unseen phishing attempts.

Researchers then explored personality and behavioral recognition to enhance detection. Tsinganos et al. [Tsinganos et al., 2018] proposed a framework that incorporated multiple elements like deception detection and speech act analysis. While this approach provided a broader understanding of social engineering attempts, the complexity of integrating multiple detection methods made real-time analysis challenging.

More advanced detection methods incorporated deep learning. Lan [Lan, 2021] introduced a bi-LSTM and ResNet model to analyze both user characteristics and chat history, improving the recognition of deception and phishing attempts. Similarly, Lee et al. [Lee et al., 2020] leveraged a BERT model to capture syntactic and semantic features, offering resilience against adversarial attacks where keywords were replaced with synonyms. However, these approaches often required significant computational power and large datasets for effective training.

A different line of research aimed at active defense mechanisms. Dalton et al. [Dalton et al., 2020] proposed an active defense pipeline that used NLP components like stylometry and dialogue engineering to waste the attacker's time, while Saleilles and Aïmeur [Saleilles and Aïmeur, 2021] created a chatbot to educate users by simulating attacks, raising awareness of SE threats. Though educational, such approaches depend on user engagement and are limited by the effectiveness of simulated interactions.

While significant progress has been made in SE detection, key challenges remain. Many methods, particularly machine learning models, are constrained by their reliance on existing datasets, making them vulnerable to novel attacks. Additionally, the high computational cost of deep learning models hampers their real-time applicability. Future research must address these limitations by improving adaptability, scalability, and efficiency across diverse SE scenarios.

1.9 Conclusion

This chapter has provided a comprehensive overview of social engineering, exploring its various facets and implications in the modern digital landscape. Among the different types of social engineering attacks, phishing stands out as particularly prevalent and damaging. Phishing attacks are a specific subset that demands close attention due to their frequency, evolving sophistication, and significant impact on both individuals and organizations.

Chapter 2

Phishing attack

2.1 Introduction

Phishing has become the weapon of choice for cybercriminals, dominating the cyber threat landscape due to its effectiveness and accessibility. Unlike attacks that exploit technical vulnerabilities, phishing targets the human element often the weakest link in cybersecurity defenses. This approach allows attackers with limited technical expertise to execute potentially devastating cyberattacks.

At its core, phishing is a form of social engineering that manipulates victims into divulging sensitive information or taking actions that compromise their security, such as harvesting login credentials, deploying malware, or gaining unauthorized access to systems. Attackers use various communication channels, including email (the most common vector), social media platforms, SMS messages (smishing), and voice calls (vishing).

Phishing techniques continue to evolve, incorporating sophisticated social engineering tactics and taking advantage of current events or trends, making the threat landscape increasingly complex. This chapter looks at the complexities of phishing attacks, exploring their different forms and the cutting-edge strategies used to detect these pervasive threats.

2.2 Phishing Attacks: An Overview

Phishing attacks are deceptive, fraudulent schemes designed to illegally obtain sensitive information or gain unauthorized access to personal, financial, or corporate data. These attacks combine social engineering techniques with technical deception, impersonating trusted entities via emails, websites, and messages. Their aim is to deceive individuals into unintentionally sharing confidential information or interacting with malicious links, ultimately compromising security. [Alkhalil et al., 2021, Gupta et al., 2016]

Attackers craft convincing imitations of legitimate sources to lure victims into believing the authenticity of the communication. The objectives may include identity theft, financial exploitation through stolen credentials, or gaining control over user information and systems.

Phishing capitalizes on human trust in electronic communication, making it a significant security threat. As a form of social engineering, it exploits psychological manipulation to bypass standard security measures.

2.3 Anatomy of a Phishing Attack

Building on the framework of the social engineering attack cycle, phishing attacks adhere to a similar structure, with nuances specifically designed to deceive victims through digital communication. The typical phishing attack follows these stages, aligning closely with the broader social engineering cycle:

- **Investigation:** Attackers identify targets by gathering information such as email addresses and social media details. This research helps craft personalized messages that increase the likelihood of success.

- **Hook:** Phishers send carefully crafted deceptive emails or messages, often impersonating trusted entities like banks or colleagues. The objective is to create urgency or fear, prompting the victim to act quickly without scrutinizing the message. This message usually contains one or more of the following: Malicious URLs leading to fake websites that mimic trusted entities. Links to download malware disguised as important documents or software updates. Requests for sensitive information under false pretenses.

- **Play:** Victims are tricked into clicking malicious links or downloading attachments that capture sensitive information or install malware. This phase exploits the trust and urgency established in the hook phase.

- **Exit:** Once the attack succeeds, phishers withdraw, often covering their tracks by using techniques like spoofing or anonymizing their digital footprint to avoid detection.

2.4 Phishing Attack Types

Phishing attacks manifest in various forms, each designed to exploit specific vulnerabilities and trick individuals into revealing sensitive information. Some common types of phishing attacks include:

- **Email Phishing :**
  This is the most prevalent form of phishing, where attackers send fraudulent emails that appear to originate from legitimate sources such as banks, online services, or colleagues. These emails often contain links to malicious websites or attachments that install malware on the recipient's device [Gupta et al., 2016].

- **Spear Phishing :**
  Unlike regular phishing, spear phishing targets specific individuals or organizations. Attackers conduct research on their targets to craft personalized and convincing emails. These emails often use information relevant to the recipient to increase the likelihood of a successful attack [Seth and Damle, 2022].

- **Whaling :**
  This is a type of spear phishing that targets high-profile individuals such as executives, government officials, or other important figures within an organization. Whaling attacks are typically highly personalized and may involve elaborate schemes to deceive the target [Alkhalil et al., 2021, P.M et al., 2023].

- **Clone Phishing :**
  In this attack, a legitimate email that has previously been sent to the victim is cloned, but with malicious links or attachments. The attacker may claim that this is a resend of an original email or an updated version [Gupta et al., 2016, Alkhalil et al., 2021].

- **Voice Phishing (Vishing) :**
  Vishing involves phone calls where attackers impersonate legitimate entities, such as banks or government agencies, to extract sensitive information from the victim. The attacker might create a sense of urgency or fear to persuade the victim to comply [Bhuvana et al., 2021].

- **SMS Phishing (Smishing) :**
  Attackers send fraudulent SMS messages that appear to come from reputable sources. These messages often contain links to malicious websites or instructions to call a number that leads to further social engineering attacks [P.M et al., 2023].

Figure 2.1 summarizes these types of social engineering attacks.

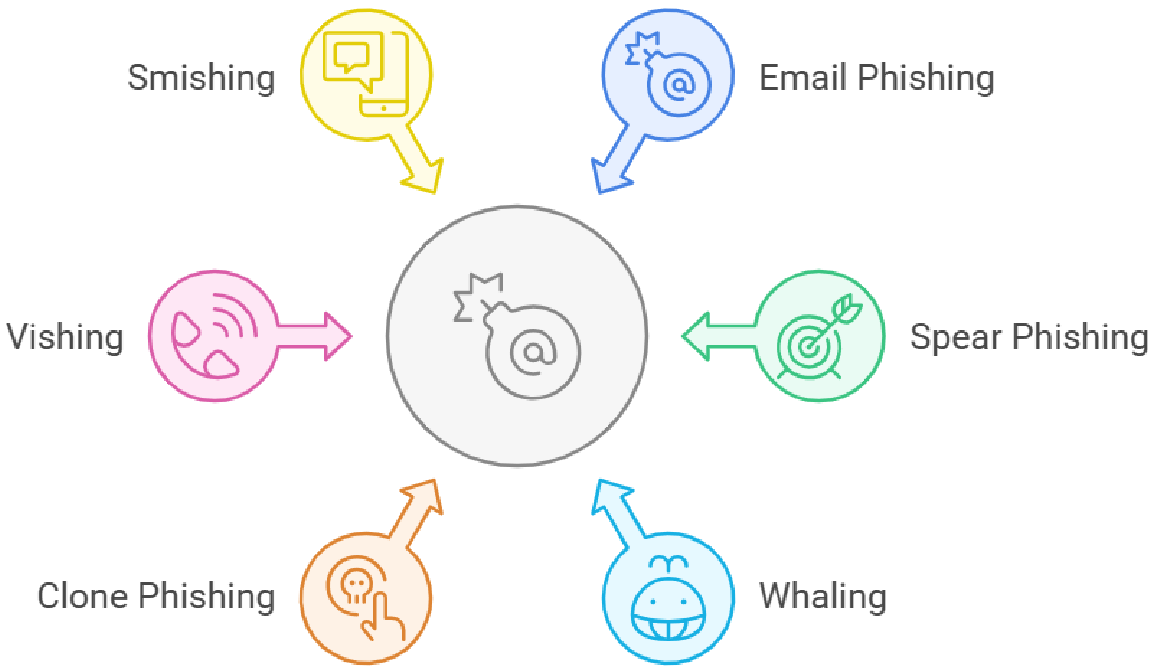


Figure 2.1: Phishing attacks type

2.5 Phishing attack impact and case study

Phishing attacks pose one of the most significant threats in today's digital landscape, affecting individuals, organizations, and even national security.

For individuals, the impact of phishing is profound, often resulting in identity theft, financial loss, and emotional distress. Victims may lose access to sensitive accounts—such as bank accounts or social media profiles—leading to unauthorized transactions and misuse of private data. The psychological effects can be severe, with individuals experiencing stress and anxiety as they struggle to regain control of their compromised information.

For businesses, the consequences of phishing attacks can be even more severe, including direct financial losses, regulatory penalties, and reputational damage. Sensitive corporate data, such as trade secrets or client information, may be compromised, exacerbating the fallout. Organizations also face operational disruptions, particularly when phishing schemes deliver ransomware, forcing system shutdowns and extensive recovery efforts.

A notable case illustrating the impact of phishing is the Twitter breach in July 2020 [Tidy, 2020]. In this incident, a group of hackers executed a sophisticated phishing scheme by creating a fake website that mimicked Twitter's internal VPN provider. They impersonated helpdesk staff, deceiving multiple employees into entering their credentials on this fraudulent site. As a result, the attackers gained access to several high-profile

Twitter accounts, including those of Barack Obama and Elon Musk. They used these accounts to solicit Bitcoin donations from followers under the pretense of doubling their contributions. Ultimately, the hackers successfully collected approximately 12.86 BTC (around $110,000 at the time) before Twitter intervened [Tidy, 2020].

2.6 Phishing Attacks Detection

This section reviews the existing research on phishing attack detection, categorizing the work into three primary areas: phishing email detection based on content, phishing detection through URL analysis, and SMS-based phishing (smishing) detection.

2.6.1 Email content

Phishing email detection based on content involves analyzing the text, headers, and other elements within the email to identify malicious intent, often using techniques like NLP alongside machine learning or deep learning models.

This category can be further divided into three subcategories based on the approach used: machine learning, deep learning, and other heuristic or rule-based methods.

2.6.1.1 Machine Learning

Egozi and Verma [Egozi and Verma, 2018] propose a linear kernel SVM that utilizes text-based features. Bountakas et al. [Bountakas et al., 2021] compare a combined model of TF-IDF and Word2Vec with various machine learning classifiers, including Random Forest, Decision Tree, Logistic Regression, Gradient Boosting Trees, and Naive Bayes. Additionally, Bountakas and Xenakis [Bountakas and Xenakis, 2023] developed an ensemble learning model that combines both textual and content features.

Kumar et al. [Kumar et al., 2020] proposed a hybrid approach that combines SVM, NLP, and probabilistic neural networks for email phishing detection. In their method, they extract distinctive features from both text and images within the emails. Feature selection is then performed on these extracted features to enhance the model's performance. To classify the emails as legitimate or phishing, they utilize a combination of SVM and probabilistic neural networks, aiming to improve detection accuracy and robustness.

2.6.1.2 Deep Learning

The authors [Fang et al., 2019] introduce a model named THEMIS for phishing email detection, building on Lai et al.'s [Lai et al., 2015] RCNN-based text classification framework. The model preprocesses data to clean up extra spaces and digital gibberish, then uses word2vec to create sequences at both word and character levels for the email header and body. These sequences are processed in parallel using Bi-LSTM, and the resulting header and body representations are combined. An attention mechanism then generates the final email representation, which is processed through a dropout layer to prevent overfitting and a sigmoid function for binary classification. Experiments using the IWSPA-AP 2018 dataset show that THEMIS outperforms other models like CNN and LSTM, achieving higher accuracy (99.848%) and a lower false positive rate (0.0043%), even on the unbalanced dataset that reflects real-world conditions.

The authors [Nguyen et al., 2018] present a two-layer architecture for detecting phishing emails using datasets from IWSPA-AP 2018, analyzing emails at both word and sentence levels. Their model first processes words into embedding vectors and applies a Bi-LSTM to capture contextual information, followed by an attention module to generate sentence representations. This process is repeated at the sentence level to create a final email body representation, which estimates the phishing probability. For emails with headers, they integrate body and header representations using similar architectures. They enhance their model, H-LSTM, with supervised attention (H-LSTM+supervised) to prioritize phishing-related words, demonstrating superior performance over a baseline SVM. Including headers improves detection results, highlighting their significance in identifying phishing emails.

The authors [Vinayakumar et al., 2018] introduce the DeepAnti-PhishNet model for detecting phishing emails across various languages. Their preprocessing involves converting text to lowercase, removing punctuation and special characters, and assigning unique identifiers to unknown words. They utilize word2vec and a neural bag of n-grams in the embedding layer to capture syntactic and semantic features of the emails. These embeddings are fed into CNN/RNN/LSTM and MLP networks to extract features, followed by a fully connected layer with a sigmoid activation function for classification. The training results using 10-fold cross-validation were lower than test results due to the small and unbalanced training dataset from IWSPA-AP 2018, causing model bias. The best model for detecting phishing emails without headers employs LSTM with word2vec, while for emails with headers, the MLP network with a neural bag of n-grams is used.

The authors [Saha et al., 2020] approach phishing detection as a classification problem with three categories: phishing websites, legitimate websites, and suspicious websites. Their model comprises three phases: data acquisition, preprocessing, and classification. Data is collected from Kaggle, then cleaned, formatted, and analyzed during preprocessing. The classification phase employs a MLP classifier, with 10-fold cross-validation yielding a 98% accuracy in training and 93% in testing. The confusion matrix indicates that detection is more accurate for legitimate and phishing websites than for suspicious ones. The authors plan to enhance detection of suspicious websites by increasing the model's complexity and incorporating a backpropagation neural network.

The authors [Barathi Ganesh et al., 2018] utilized fastText for word embedding, which captures word representations and their predicted classes. They trained two models using datasets from the IWSPA-AP 2018: one model handled emails with and without headers separately, while the other combined both types into a single model. Results indicated that the independent model for separate email types outperformed the combined model. They concluded that fastText's ability to provide semantic representation with low computational cost and reliable performance makes it effective for real-world data.

The authors [Abutair et al., 2019] propose a method for phishing detection that involves two main steps: data preprocessing and classification. In the preprocessing step, a dataset is generated using dialogue preprocessing, incorporating features like link score (malicious link detection), spelling score (spelling quality), and intent score (intent detection using a blacklist), each scored from 0 to 1. For classification, an artificial neural network (MLP) is used to determine if the input indicates an attack (high value). Additionally, they employ an ensemble learning approach with a Gaussian Naive Bayes classifier, Decision Tree, and Random Forest using soft voting as an alternative to the MLP classifier. Experimental results show that both MLP and ensemble learning achieve comparable accuracy, but the ensemble method with soft voting outperforms MLP in terms of f1-score and AUC.

2.6.1.3 Others

In [Giorgi et al., 2020] analyses sender-receiver interactions in spearphishing emails in order to facilitate end-to-end authorship verification. Their study proposes two scenarios for email authorship verification: sender verification and end-to-end verification. They propose a binary classification method for authorship verification that uses both standard machine learning classifiers and deep learning models. A dataset of approximately 150 emails from Enron employees was used for the experiments. The dataset contains a variety of lengths, with most of the emails being long, followed by medium-length messages, and a small number of short messages. The results indicate that end-to-end email verification provides greater accuracy compared to traditional sender verification methods.

The authors of [Seifollahi et al., 2017] present an innovative approach to authorship analysis of phishing emails. They combine term frequencies (TFs) with a path-similarity distance (PSD) measure. An algorithm is developed to detect clusters of similar emails in phishing datasets, combining clustering and feature selection techniques. In addition to the well-known K-means algorithm, three optimization-based clustering algorithms are used in the study: DCClust, INCA, and MS-MGKM.

2.6.1.4 Synthesis

Phishing email is a serious problem that can affect anyone, with the goal of stealing sensitive information or gaining access to the victim's internal network. As a result, there was a lot of research in this area, but most of it focused on phishing URLs rather than email text. The most popular approach is the blacklist, but it has some drawbacks, such as its inability to detect new phishing URLs and the malicious URL's constant change, which Sheng et al. [Sheng et al., 2009] investigated.

With the increasing use of machine learning in many domains, researchers have attempted to apply it in phishing detection, where there is a lot of work, using techniques such as K-Nearest Neighbor (KNN) [Zamir et al., 2020], Naive Bayes (NB) [Alqahtani et al., 2020], and SVM [Figueroa et al., 2017]. However, this method necessitates manual engineering features, which necessitate domain expertise and manual labor.

These issues, as well as the rapid development of deep learning, is the cause that moves the researcher to use this approach. Fang et al. [Fang et al., 2019] propose a new representation employing a fourth different level header and body, where each is represented in character and word level, as well as an attention mechanism with an improved RCNN. Nguyen et al. [Nguyen et al., 2018] also propose hierarchical LSTM with a supervised attention mechanism for the representation of the email at the word and sentence levels. Douz et al. [Douzi et al., 2017] proposed a new method for detecting phishing email text and phishing URLs using simple voting. Barathi Ganesh et al. [Barathi Ganesh et al., 2018], on the other hand, propose using a new word embedding (fastText) to obtain the word representation as well as it's class. Table 2.1 summarises the related work.

Tableau 2.1: Summary of Related Work in Phishing Detection [Remmide et al., 2024c]

Authors	Classification approach	Classifier	Feature extraction
Alam et al.	Machine learning	Random Forest	PCA
Egozi and Verma	Machine learning	Linear-SVM	Static features
Bountakas et al.	Machine learning	Random Forest	TF-IDF and Word2Vec
Bountakas and Xenakis	Machine learning	Ensemble Learning	hybrid features with word2ve
Barathi Ganesh et al.	Deep learning	fastText	fastText
Nguyen et al.	Deep learning	LSTM	Hierarchical LSTMs
Halgas et al.	Deep learning	RNN	Tokenized text
Alhogail and Alsabih	Deep learning	GCN	TF-IDF
Douz et al.	Deep learning	Linear regression	Autoencoder
Fang et al.	Deep learning	RCNN	Word2vec
Lee et al.	Deep learning	BERT	BERT
Li et al.	Deep learning	LSTM	custom feature

2.6.2 URL

Phishing detection based on URLs focuses on analyzing links embedded in emails or messages to determine their legitimacy. This can involve a range of techniques, including blacklists, heuristic analysis, and machine learning classifiers to identify suspicious or malicious URLs.

We have further divided phishing detection based on URLs into two subcategories according to the approach used: machine learning and deep learning.

2.6.2.1 Machine learning

[Gitelman, 2014] proposed an SVM-based technique for detecting phishing URLs. The method utilizes features such as the structure of the URL, the presence of specific words, and brand names within the URI.

[Hai and Hwang, 2018] introduced a machine learning approach for URL classification, leveraging natural language processing features with word vector representation and n-gram models as key features. They used features extracted from n-gram models, word vector representations, and other lexical properties to build a classification model using the SVM algorithm. The model utilized 100 features from the word2vec representation. Among the 150,397 URLs analyzed, 107,615 were benign and 42,782 were malicious. The SVM achieved an accuracy rate of 97.1% and an F1 score of 0.95, with a classification time of just 0.01 seconds.

[Sahingoz et al., 2019] developed a machine learning-based technique for phishing URL detection, employing seven different classification algorithms and features derived from NLP. Their experimental results demonstrated that the Random Forest (RF) algorithm, using NLP-based features, achieved a high accuracy of 97.98% in detecting phishing URLs.

2.6.2.2 Deep learning

[Wei et al., 2020] proposed a method for identifying phishing websites using a deep neural network with convolutional layers that analyzes the text of the URL address. They encoded URLs as one-shot character-level vectors and used these as inputs to a convolutional neural network. Their approach demonstrated that a dictionary-free analysis of URLs can achieve nearly 100% accuracy in detecting phishing sites, offering effective zero-day defense. The trained phishing detector is compact and efficient, making it suitable for deployment on mobile devices.

[Rasymas and Dovydaitis, 2020] proposed a method for detecting phishing sites using only their URLs. They employed a deep neural network architecture to classify phishing and benign URLs, achieving an accuracy of 94.4%. Various feature combinations were evaluated, including lexical features, character-level embeddings, word-level embeddings, and combinations thereof. The highest accuracy was attained using a model that combined character and word embeddings.

[Huang et al., 2019] introduced PhishingNet, a deep learning framework designed to effectively detect phishing URLs. PhishingNet combines a character-level convolutional neural network module with an attention-based hierarchical recurrent neural network (RNN) module. These parallel modules extract comprehensive URL feature representations. The framework achieved a precision of 0.98 and an accuracy of 0.97.

[Le et al., 2018] developed URLNet, an end-to-end deep learning method for detecting phishing URLs. They applied a convolutional neural network to both characters and words within the URL string, training the model through a mutually optimized embedding process. Their results demonstrated that URLNet outperformed existing models in phishing URL detection. However, the model may be less effective when dealing with phishing sites that use very short URLs.

Tableau 2.2: A comparison of related phishing URL detection work

Research	Technique	Method	Dataset	Results
[Bahnsen et al., 2017]	Deep learning	LSTM	1 million URLs	98.70% precision
[Hai and Hwang, 2018]	Machine learning	SVM	150 397 URLs (107 615 legitimes, 42 782 phishings)	95% precision
[Abutair et al., 2019]	Machine learning	CRB-PDS	D1 : 500 URLs D2 : 750 URLs	95% precision
[Sahingoz et al., 2019]	Machine learning	KNN	11 055 URLs	97.98% precision
[Wang et al., 2019]	Deep learning	PDRCNN	500 000 URLs from PhishStack	97% precision
[Rasymas and Dovydaitis, 2020]	Deep learning		387 772 URLs (153 141 phishings, 234 631 legitimes)	94.4% precision
[Wei et al., 2020]	Deep learning	CNN	21 208 URLs	99.98% precision

2.6.2.3 Synthesis

The detection of phishing URLs has seen significant advancements through the application of both machine learning and deep learning techniques. In the realm of machine learning, researchers have explored various approaches to enhance detection accuracy and efficiency. Notably, Sahoo et al. [Sahoo et al., 2017] proposed an SVM-based technique that leverages the URL structure, specific words, and brand names within the URI as key features. Expanding on this, Hai et al. [Hai and Hwang, 2018] introduced a more sophisticated approach using natural language processing features, incorporating word vector representations and n-gram models. Their SVM-based model achieved an impressive accuracy of 97.1% and an F1 score of 0.95, demonstrating the potential of combining lexical properties with advanced feature extraction techniques. Sahingoz et al. [Sahingoz et al., 2019] further advanced the field by employing seven different classification algorithms, with their Random Forest model achieving a high accuracy of 97.98% using NLP-based features.

The application of deep learning techniques has pushed the boundaries of phishing URL detection even further. Wei et al. [Wei et al., 2020] proposed a novel approach using a deep neural network with convolutional layers to analyze URL text, achieving nearly 100% accuracy and offering effective zero-day defense. This method's efficiency makes it suitable for mobile device deployment. Rasymas et al. [Rasymas and Dovydaitis, 2020] explored various feature combinations, including lexical features and both character-level and word-level embeddings, achieving 94.4% accuracy with a model combining character and word embeddings. Huang et al. [Huang et al., 2019] introduced PhishingNet, a framework that combines a character-level CNN with an attention-based hierarchical RNN, achieving high precision (0.98) and accuracy (0.97). Le et al. [Le et al., 2018] developed URLNet, an end-to-end deep learning method applying CNNs to both characters and words within the URL string, which outperformed existing models but showed limitations with very short URLs. These deep learning approaches demonstrate the potential for highly accurate, efficient, and adaptable phishing URL detection systems, capable of capturing complex patterns and relationships within URL structures.

2.6.3 Smishing

A rule-based strategy to recognize smishing communications has been proposed by [Jain and Gupta, 2018]. To separate smishing messages from authentic messages, they set up nine rules. Furthermore, these rules have been taught using various categorization methods, including PRISM, RIPPER and the Decision Tree. In the performance assessment of the approach, more than 99% of actual negative messages are negative.

[Mishra and Soni, 2019] proposed a content-based method to identify smishing messages in another work that was suggested. The most common words used in smishing SMS are determined using a machine learning system. In addition, this template evaluates the appearance of the login page and the download of the.apk file to check if the URL is malicious.

A machine learning-based algorithm to identify smishing messages was proposed by [Balim and Gunal, 2019]. The model classifies communications as valid or as smishing messages using a combination of feature engineering and machine learning algorithms. During the characteristics engineering process, the relevant characteristics of the messages are extracted, such as the existence of specific words or phrases. The model uses decision trees, random forests and support vector machines as machine learning algorithms. The model was trained and tested on a set of authentic message data and smishing messages.

[Sonowal, 2020] used four correlation techniques to rank the characteristics in the most recent study effort for smishing detection, including Pearson rank corrélation, Spearman rank correllation, Kendall rank correlation, and Point biserial rank correlation. The optimal set of characteristics is finally chosen to identify smishing with a 98.40

A two-stage smishing detection system – domain verification and SMS classification – has been successfully created and tested by [Mishra and Soni, 2021]. To assess the maliciousness of the SMS, each phase focused on a different part of the message. They are mainly focused on reviewing the legitimacy of the URL in the SMS while minimizing the complexity of the system. The Backpropagation approach was used to develop the system, and the resulting accuracy was 97.93

[Boukari et al., 2021] introduced a machine-learning-based technique for smishing fraud detection. Based on the results of the tool, they found that Naive Bayes is a better classifier when the false-negative score is taken into account, even though Random Forest produces better measurement measurements. the resulting accuracy was 90.59% with the Naive bayes algorithm, and 98.15% with Random forest.

A smishing detection technique that combines a text classifier and an URL classifier has been proposed by [Jain et al., 2022a].The message was analyzed by the template, and if an URL was found, it was sent to a URL classifier. Voting classifiers are used in the proposed method to determine whether a message is smishing or not. Voting classifiers combine the results of different models. The proposed approach produces a 98.94% accuracy rate.

[Mambina et al., 2022] proposed a machine learning-based methodology to detect Smishing. The best model, with an accuracy of 99.86%, is a hybrid model using Extratree characteristic selection and Random Forest using TFIDF vectorization (Term Frequency Inverse Document Frequency). The results are compared to a Naive Bayes multinomial model as a reference. A database of 32259 messages in swahili is used to evaluate performance.

[Zhou et al., 2015] used a C-LSTM model for classifying feelings and questions from a specified database set. CNN and RNN are merged. RNN is used to create sentences from the sentences recovered once the phrases are extracted using CNN.

[Roy et al., 2020] used a deep learning model based on CNN and LSTM for classifying spam communications in the research study. Their methodology is put into action in three stages: first, a word matrix is produced, then characteristics are identified, and finally, classification is carried out in the third step. The accuracy is 99.44%.

[Shravasti and Chavan,] proposed a model for smishing detection through deep learning. The proposed approach is tested using several classification models. The LSTM, KNeighbors, Stochastic Gradient Descent (SGB), Decision Tree, Naive Bayes and Random Forest Classifiers are examples of classification models used. The LSTM gives better results than other classifiers. The accuracy of the model is 95.11%.

[Mishra and Soni, 2022] used a neural network to create an effective smishing detection system. In addition, using a neural network, the seven main features of smishing SMS are extracted. The accuracy obtained is compared to the results of the classification of machine learning algorithms. With a difference of 1.11%, the comparison shows that neural networks produced more accurate results.

A heuristic-based technique to identify smishing messages has been proposed by [Jain and Gupta, 2019]. The authors used classification methods to rank messages based on the 10 characteristics they had chosen in smishing messages. The authors tested their strategy using a set of manually edited data. The results The accuracy ratio was 98.74%.

[Sheikhi et al., 2020] used mediated neural networks with a hidden layer for Create a precise template for recognizing SMS spam messages based on content-based criteria. The evaluation results indicate that the extracted characteristics have a strong association with the message class and that the medium neural network technique is able to accurately classify the messages class with a high F measurement rate. The results indicate a 98.8% accuracy rate.

[Goel and Jain, 2018] have proposed to use a technique called "smishing classifier" for Listing of smishing messages. The three phases of this methodology are SMS analysis, SMS standardization and SMS classification phase. After studying the URL contained in the message during the SMS analysis phase, the message is dealt with in more detail. Lastly, the SMS classification step allows us to classify SMS using the Naive Bayes classification algorithm. The SMS standardization phase normalizes, or converts the text into root, the text present in the SMS. The proposed frame searches for the sender's URL and mobile phone number in the blacklists.

In their study, the authors [Chen et al., 2023] propose the use of a machine learning model for the purpose of analysing the contents of messages and identifying URLs associated with phishing activities. In order to tackle the problem at hand, the researchers gathered real text messages from mobile phones, preserving their anonymity, and proceeded to manually categorise them. Subsequently, they employed machine learning algorithms such as SVM, MultinomialNB, RF, and Logistic Regression to effectively detect fraud. The approach proposed by the authors achieved an F1 score of 94%.

The authors of the study [Karhani et al., 2023] propose a machine learning model that combines domain-related features with Decision Tree and NLP methods to accurately detect phishing and smishing attacks. By training the model on smishing messages, from TELUS Corporation and two public dataset [The National University of Singapore SMS Corpus:, 2023, Shahrivari et al., 2020] they were able to achieve an accuracy of 99.40% and an F1 score exceeding 99%.

[Mishra and Soni, 2022] compared machine learning algorithms and neural networks. Their comparison revealed that neural networks exhibited superior accuracy, with a difference of 1.11%.

[Boukari et al., 2021] concluded that Naive Bayes is a better classifier when considering the false negatives score, even though Random Forest produces better overall results.

Table 7.2 and Figure 2.2 provide a comprehensive summary of the related works examined in this study on Smishing detection. As it can be seen the most widely used technique for detecting SMS phishing is machine learning, while deep learning is rarely employed.

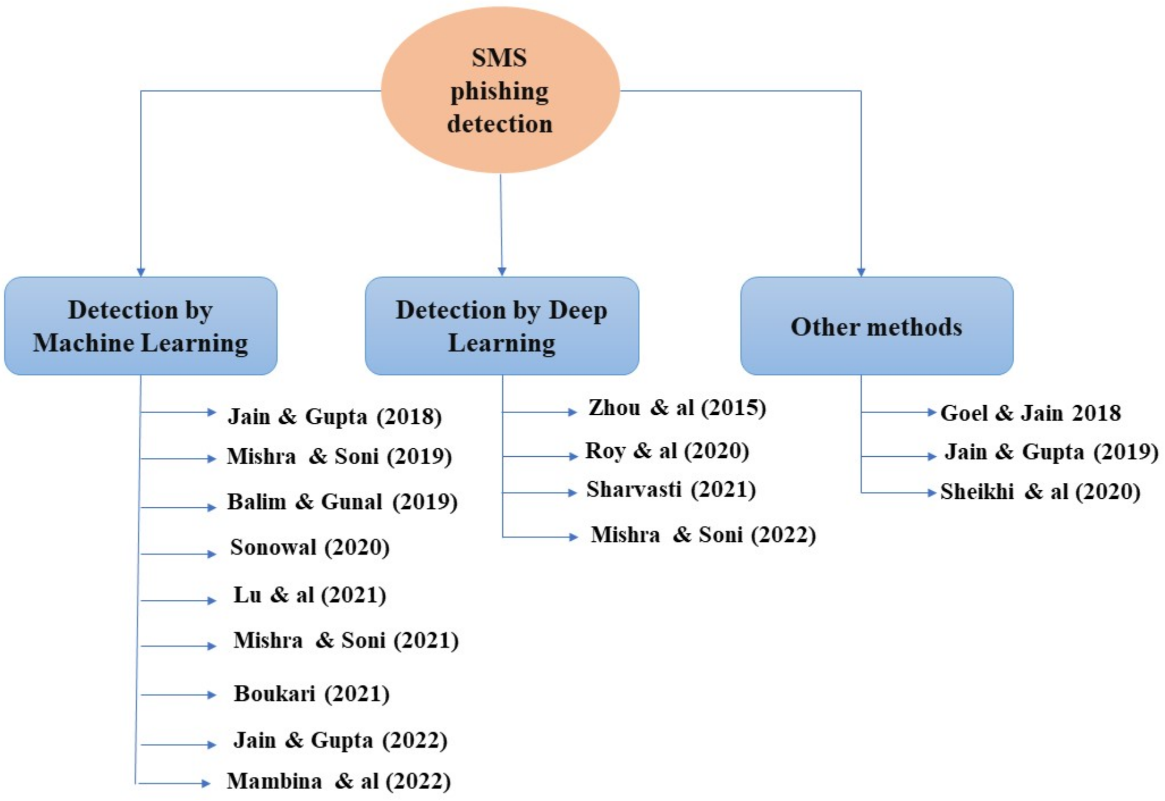


Figure 2.2: Overview of smishing detection methodologies

Tableau 2.3: A comparison of related works on SMS phishing detection [Remmide et al., 2024d]

Research	Technique	Method	Dataset	Results
[Zhou et al., 2015]	Deep Learning	CNN and RNN	11855 SMS	87.8%
[Roy et al., 2020]	Deep Learning	CNN and LSTM	5574 SMS	99.4%

Continued on next page

Research	Technique	Method	Dataset	Results
[Shravasti and Chavan,]	Deep Learning	LSTM, KNeighbors, SGD, Decision Tree, Naive Bayes, Random Forest Classifier	5572 SMS	95.11%
[Mishra and Soni, 2022]	Neural Network	ANN	5858 SMS	97.91%
[Jain and Gupta, 2018]	Machine Learning	PRISM , RIPPER, and Decision Tree	5574 SMS	99%
[Mishra and Soni, 2019]	Machine Learning	Naive Bayes Classifier, Random Forest Classifier, Decision Tree Classifier	5572 SMS	96.29%
[Balim and Gunal, 2019]	Machine Learning	Decision Trees, Random Forests, Support Vector Machines	/ SMS	/
[Sonowal, 2020]	Machine Learning	Random Forest, Decision Tree Classifier, AdaBoost Classifier, Support Vector Machine	5578 SMS	98.40%
[Liu et al., 2021]	Machine Learning	Logistic Regression	31.97 million SMS	96.16%
[Mishra and Soni, 2021]	Machine Learning	Backpropagation Algorithm, Random Forest, Naive Bayes, Decision Tree	5858 SMS	97.93%
[Boukari et al., 2021]	Machine Learning	Naive Bayes, Random Forest	5000 SMS	98.15%
[Jain et al., 2022a]	Machine Learning	XGB , GBDT , RF, BgC , KNN, ETC , DT, LR, AdaBoost, BNB, MNB , SVC , and GNB	5179 SMS, 507195 URL	98.94%
[Chen et al., 2023]	Machine Learning	SVM, MultinomialNB, RF, and Logistic Regression	557 SMS	F1 score of 94%

Continued on next page

Research	Technique	Method	Dataset	Results
[Karhani et al., 2023]	Machine Learning	hybrid between DT and SVC	143,623 SMS	99.40%
[Mambina et al., 2022]	Machine Learning	Random Forest, Naive Bayes, SVM, KNN, Adaboost, Logistic Regression, Extra Tree Classifier	32259 SMS	99.86%
[Goel and Jain, 2018]	Blacklist	Naive Bayesian Classifier, Bayesian Classifier	/	/
[Jain and Gupta, 2019]	Heuristic	SVM, Logistic Regression, Neural Network, Naive Bayes, Random Forest	5574 SMS	98.74%
[Sheikhi et al., 2020]	Average Neural Network	/	5574 SMS	98.8%

2.6.3.1 Synthesis

The literature on smishing (SMS phishing) detection reveals a diverse array of approaches, predominantly leveraging machine learning techniques, with an emerging interest in deep learning methodologies. Early strategies employed rule-based systems, such as the work by Jain et al. [Jain and Gupta, 2018], which relied on predefined rules combined with classification algorithms. However, the field has rapidly evolved towards more sophisticated machine learning approaches. These include feature engineering techniques, as demonstrated by Balim et al. [Balim and Gunal, 2019] and Sonowal et al. [Sonowal, 2020], and the application of classical algorithms such as Naive Bayes, Random Forest, and Support Vector Machines. Some researchers, like Jain et al. [Jain et al., 2022a], have proposed ensemble methods using voting classifiers to improve performance. Content-based analysis, focusing on common words in smishing messages and URL evaluation, has also been a significant area of research, as seen in the work of Mishra et al. [Mishra and Soni, 2019].

While machine learning dominates the field, deep learning approaches are gaining traction, showing promising results that often outperform traditional methods. Researchers like Mishra et al. [Mishra and Soni, 2022] and Sheikhi et al. [Sheikhi et al., 2020] have utilized neural networks, while others such as Zhou et al. [Zhou et al., 2015] and Roy et al. [Roy et al., 2020] have proposed hybrid models combining CNN and LSTM networks. Multi-stage approaches, such as the two-stage system by Mishra et al. [Mishra and Soni, 2021] involving domain verification and SMS classification, have also emerged. Performance metrics across these studies are impressive, with reported accuracies ranging from 90.59% to 99.86%, and many achieving over 98% accuracy. The field shows trends towards hybrid models, increased focus on feature importance, and exploration of multi-lingual approaches. As smishing tactics continue to evolve, future research may focus on improving the adaptability of detection systems, enhancing multi-lingual capabilities, and further exploring the potential of deep learning in this critical domain of cybersecurity.

2.7 Conclusion

In this chapter, we have provided a comprehensive overview of phishing attacks, detailing their anatomy, various types, and the current state-of-the-art detection techniques. We explored how these attacks exploit the human element, often the weakest link in cyber-security, and highlighted the diverse communication channels used by cybercriminals to execute their schemes.

As we transition to the next chapter, which focuses on the theoretical background of case-based reasoning and deep learning, it is crucial to consider how these advanced methodologies can enhance our defenses against phishing. By leveraging machine learning and AI-driven approaches, we can develop robust detection systems capable of identifying and mitigating phishing threats in real-time but also evolve in response to emerging tactics. This integration of technology will be crucial in fortifying our cybersecurity posture and addressing the complexities of phishing in an increasingly digital world.

Chapter 3

Deep learning and case-based reasoning

3.1 Introduction

Artificial intelligence (AI) aims to solve complex problems through advanced computational methods. Within AI, deep learning has become a key technique, using multi-layered neural networks to automatically extract features and discover complex patterns in large datasets. It has transformed fields like computer vision, natural language processing, and speech recognition by eliminating the need for manual feature engineering.

In contrast, case-based reasoning (CBR) is a problem-solving approach that draws on past cases to solve new ones. By retrieving and adapting solutions from similar problems, CBR offers a more experience-driven, human-like reasoning process. It has been applied in various domains, including phishing detection, where historical data informs the identification of new threats.

This chapter will begin by outlining the principles and architectures of deep learning, followed by an exploration of CBR and some examples of AI applications.

3.2 Deep Learning

Deep learning is a subfield of machine learning that focuses on artificial neural networks, which are modeled after the structure and function of the human brain. While traditional machine learning techniques often require manual feature engineering, deep learning models can automatically discover the most relevant features from raw data, such as images, texts, or audio files.

The deep learning model is composed of multiple interconnected layers of artificial neurons, each performing a specific transformation on the input data. As the data flows through these layers, the model learns to extract progressively more complex and abstract representations, enabling it to tackle a wide range of tasks with remarkable accuracy.

One of the key advantages of deep learning is its ability to achieve human-level or even superhuman performance on various cognitive tasks. By training on large, labeled datasets and leveraging the expressive power of deep neural network architectures, deep learning models can learn intricate patterns and relationships that were previously challenging for traditional algorithms to capture.

The training process of deep learning models typically involves backpropagation, a technique that efficiently propagates the error signals backward through the network layers, allowing the model to adjust its internal parameters and improve its predictive capabilities. This iterative process of learning from data continues until the model reaches the desired level of performance.

Furthermore, the versatility of deep learning has led to its widespread adoption across diverse domains, from computer vision and natural language processing to speech recognition and bioinformatics. Researchers and practitioners continue to push the boundaries of deep learning, developing novel architectural designs, training strategies, and applications to tackle increasingly complex problems.

3.3 Convolutional neural networks (CNNs)

A Convolutional Neural Network (CNN) [Huang et al., 2019] is a deep learning model designed to process data with a grid-like structure, such as images. Inspired by the organization of the animal visual cortex, CNNs are designed to automatically and adaptively learn spatial hierarchies of features, capturing patterns from lower-level details to more abstract, high-level representations. CNNs are composed of three main types of layers that work together to form the overall architecture:

- Convolution layers, which are responsible for feature extraction by applying filters to the input data, detecting features like edges or textures.
- Pooling layers, which downsample the feature maps, reducing dimensionality and retaining important information.
- Fully connected layers, which combine the extracted features and map them to the final output, such as classification results.

Together, these layers enable CNNs to perform tasks such as image classification and object recognition with remarkable accuracy by learning progressively complex representations of the input data.

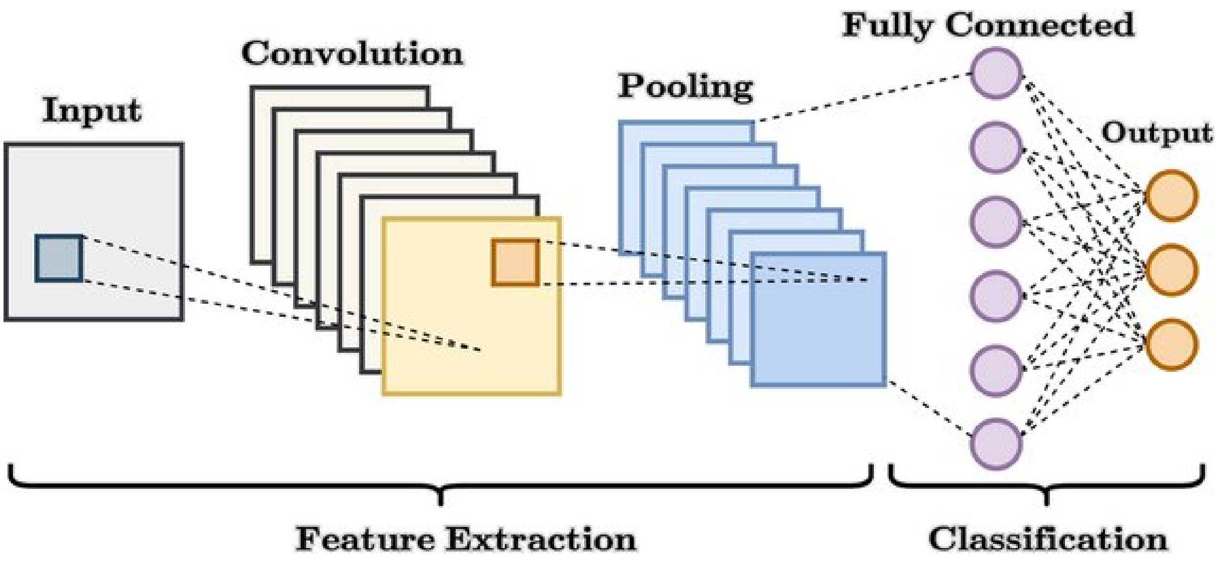


Figure 3.1: Architecture of CNN [Das et al., 2024]

Convolution layer

The purpose of the convolution operation in a CNN is to extract key features from the input data, such as edges, colors, and gradient orientations. Typically, the first convolutional layer (ConvLayer) captures low-level features, while deeper layers progressively learn to detect high-level features, allowing the network to develop a more comprehensive understanding of the images in the dataset.

At the core of the convolutional process is the kernel or filter, a small matrix that slides over the input data. The kernel is responsible for tasks like blurring, sharpening, and edge detection by performing a pointwise operation (a dot product) between the kernel and a portion of the input image. This operation generates feature maps that highlight important patterns within the image, enabling the network to learn and recognize these patterns as it moves through deeper layers.

Pooling layer

The purpose of pooling layers [Scherer et al., 2010] is to introduce spatial invariance by reducing the resolution of the feature maps, making the network more robust to variations like translation and scale. Each pooled feature map corresponds to a feature map from the previous layer, but with reduced dimensionality, allowing the network to retain important information while minimizing computational complexity.

There are several types of pooling operations, with the most common being max pooling and average pooling (sub-sampling).

- Max pooling selects the maximum value from a defined neighborhood, effectively preserving the most prominent features.
- Average pooling computes the average of the values within the kernel's receptive field, providing a more generalized representation of the region.

By reducing spatial dimensions, pooling layers help maintain key patterns while discarding less significant details, contributing to the overall efficiency and performance of the network.

Fully-connected layer

The fully connected layer in a neural network functions like a traditional neural network, where each neuron connects to every neuron in the previous layer. Its primary role is to transform the high-level features extracted by earlier layers into final predictions, such as class scores or probabilities. By computing the scores for each output class and applying an activation function like softmax, the fully connected layer converts these scores into probabilities, enabling the model to make a final classification decision. It serves as the final stage where all learned patterns are consolidated into actionable outcomes.

3.4 Recurrent neural networks (RNNs)

A Recurrent Neural Network (RNN) is a powerful sequence model well-suited for tasks such as language modeling, speech recognition, and machine translation [Zaremba et al., 2014]. However, it is widely recognized that effective regularization is crucial for the success of neural network applications. The most commonly used regularization method for feedforward neural networks, dropout, does not perform as effectively in RNNs. Due to the tendency of large RNNs to overfit, many practical applications rely on smaller models, limiting their potential. Current regularization techniques provide only modest improvements for RNNs, highlighting a need for more advanced approaches to prevent overfitting in these models [Zaremba et al., 2014].

The general architecture of the RNN is shown in Figure 3.2:

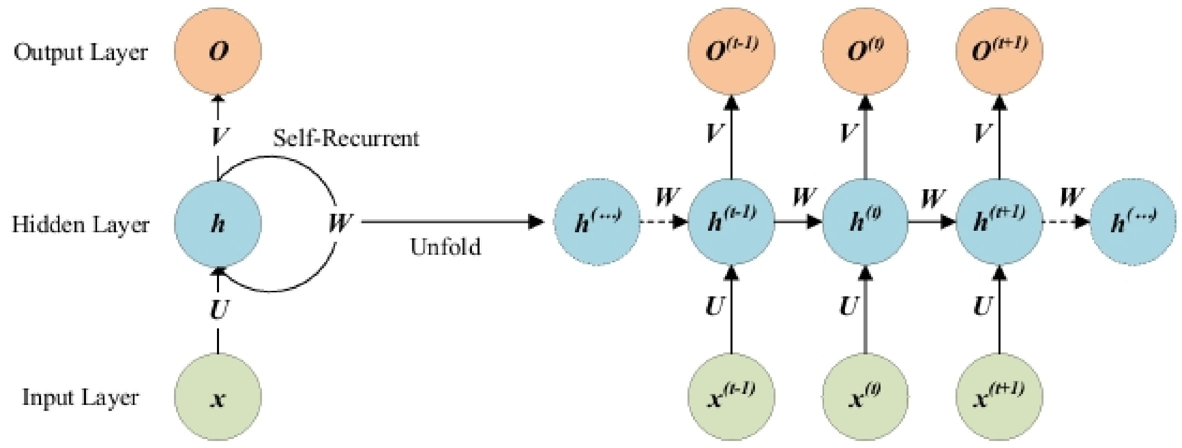


Figure 3.2: Overview of the Recurrent Neural Network (RNN) Architecture [Yin et al., 2021]

3.5 Long Short-Term Memory(LSTM)

The LSTM is a specific type of recurrent neural network. This architecture was introduced primarily to address the issue of vanishing and exploding gradients. This form of network is also better at maintaining long-distance connections and understanding the relationship between values at the start and end of a sequence [Alawneh et al., 2020].

The LSTM model introduces expressions, namely gates. There are actually three different types of gates:

- Forget gate: regulates the amount of information received by the memory cell from the previous phase.
- Update (input) gate: determines whether the memory cell should be updated or not. It also regulates the amount of data that any new memory cell will send to the current memory cell.
- Output gate: controls the value of the next hidden state.

Figure 3.3 shows the architecture of the LSTM block:

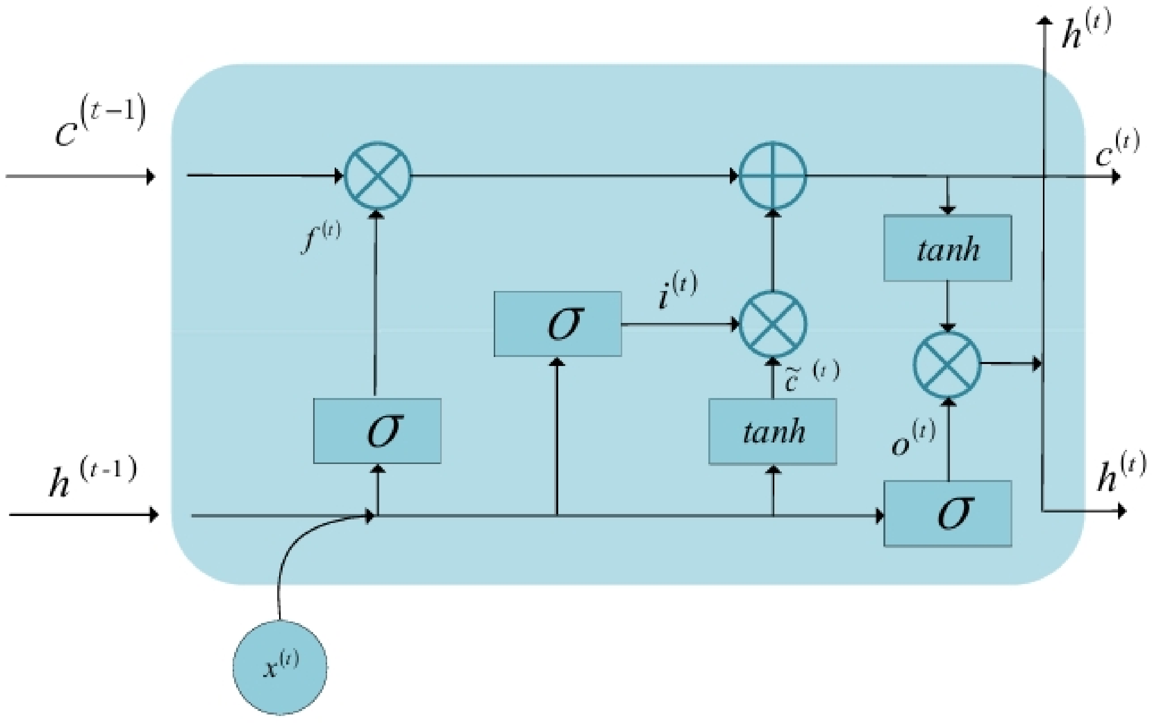


Figure 3.3: Architecture of the LSTM block [Yin et al., 2021]

3.6 Bi-LSTM

A widely used variant of recurrent neural networks for natural language processing is the Bidirectional Long Short-Term Memory (BiLSTM) network. Unlike a standard LSTM, which processes input in one direction, BiLSTM captures information from both past and future contexts by allowing the input to flow in both directions. This makes it particularly effective for modeling sequential relationships in text, such as dependencies between words and sentences [Alawneh et al., 2020].

BiLSTM achieves this bidirectional flow by introducing an additional LSTM layer that processes the input sequence in the reverse order. The outputs from the forward and backward LSTM layers are then combined using techniques like averaging, summation, multiplication, or concatenation, enhancing the model's ability to understand complex sequence relationships [Alawneh et al., 2020].

The architecture of Bi-LSTM is shown in Figure 3.4 :

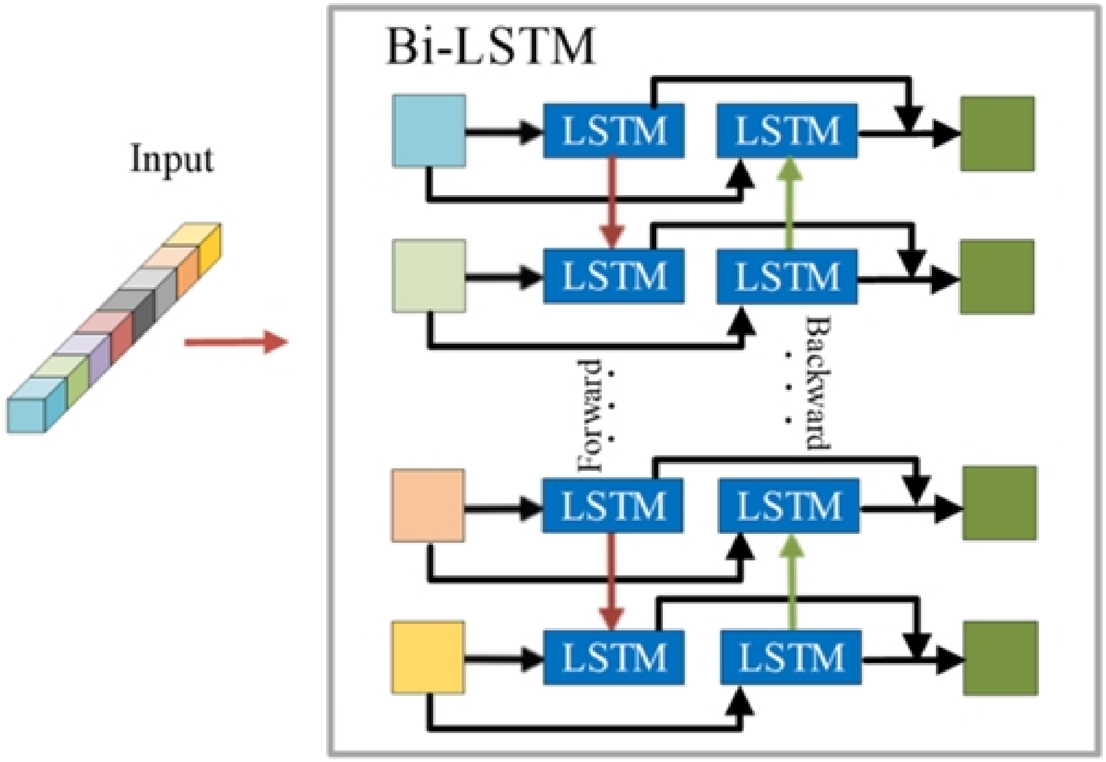
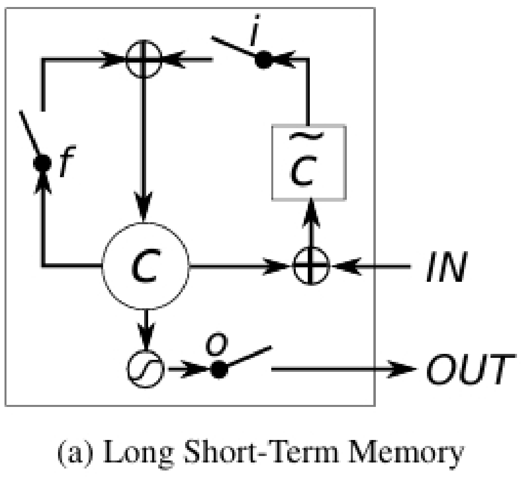


Figure 3.4: Architecture of the BiLSTM [Yin et al., 2021]

3.7 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU), introduced by Cho et al. in 2014 [Chung et al., 2014], is an evolution of the standard RNN architecture. Designed to address the limitations of traditional RNNs, such as the vanishing gradient problem, GRUs simplify the LSTM network while retaining its ability to capture long-term dependencies in sequential data. By reducing the complexity of LSTMs, GRUs offer a computationally efficient alternative that performs well in a variety of tasks, including time-series prediction, NLP, and speech recognition.

Traditional RNNs struggle with long-range dependencies due to their inability to propagate gradients effectively through extended sequences. LSTMs addressed this issue by introducing memory cells and gating mechanisms, but their architecture involves three gates and multiple internal states, which increases computational overhead. GRUs simplify this structure by using only two gates—the update gate and the reset gate—to manage the flow of information, making them more efficient while still effectively capturing dependencies over time. Figure 3.5 illustrates the key differences between the two models



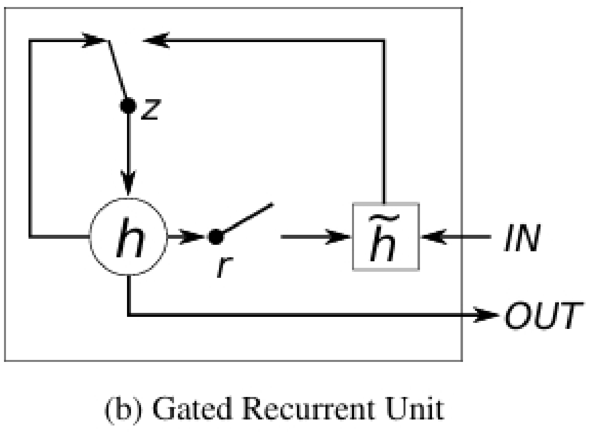


Figure 3.5: Illustration of LSTM and GRU [Chung et al., 2014]

The GRU architecture revolves around these two gates, which selectively control the information retained or discarded at each time step. The update gate (z_t) determines how much of the previous hidden state is carried forward to influence the current hidden state, while the reset gate (r_t) controls how much of the past information should be forgotten. This allows GRUs to adaptively capture dependencies of varying lengths, making them particularly well-suited for tasks involving complex sequential data.

Mathematically, the update and reset gates are computed using sigmoid functions of the previous hidden state and the current input. The candidate hidden state is calculated using the tanh activation function, incorporating the output of the reset gate. The final hidden state is a weighted combination of the previous hidden state and the candidate hidden state, with the update gate controlling the weights. This formulation ensures efficient information flow through the network, allowing GRUs to learn and represent intricate temporal patterns while mitigating the vanishing gradient problem.

GRUs have gained widespread popularity due to their balance of simplicity and performance. In NLP, they have been successfully applied to tasks such as machine translation, text classification, and sentiment analysis. Beyond NLP, GRUs are used in time series analysis, speech recognition, and even in bioinformatics, for tasks like gene expression modeling and protein structure prediction.

While GRUs are highly effective, there are cases where LSTMs outperform them, particularly in tasks involving more complex temporal dynamics. LSTMs' more elaborate gating mechanisms provide greater flexibility for managing long-range dependencies, making them better suited for such challenges.

3.8 Temporal convolutional network (TCN)

Temporal Convolutional Networks (TCNs) represent a significant advancement in neural network architecture, specifically designed for processing sequential data [Bai et al., 2018]. TCNs utilize specialized convolutions and dilations to effectively handle temporal data while maintaining large receptive fields, making them particularly suitable for sequence-to-sequence tasks.

Causal convolutions

TCNs are distinguished by two fundamental characteristics. First, the network ensures that output at time t depends only on inputs from time t and earlier, never on future inputs [Lara-Benítez et al., 2020]. Second, TCNs can process time series of any length and map them to outputs of identical length, similar to RNNs. To implement these features, TCNs employ one-dimensional convolutional networks in the first layer, with zero padding to maintain consistent layer sizes. They also utilize dilated convolutions to achieve long effective histories without excessive computational burden.

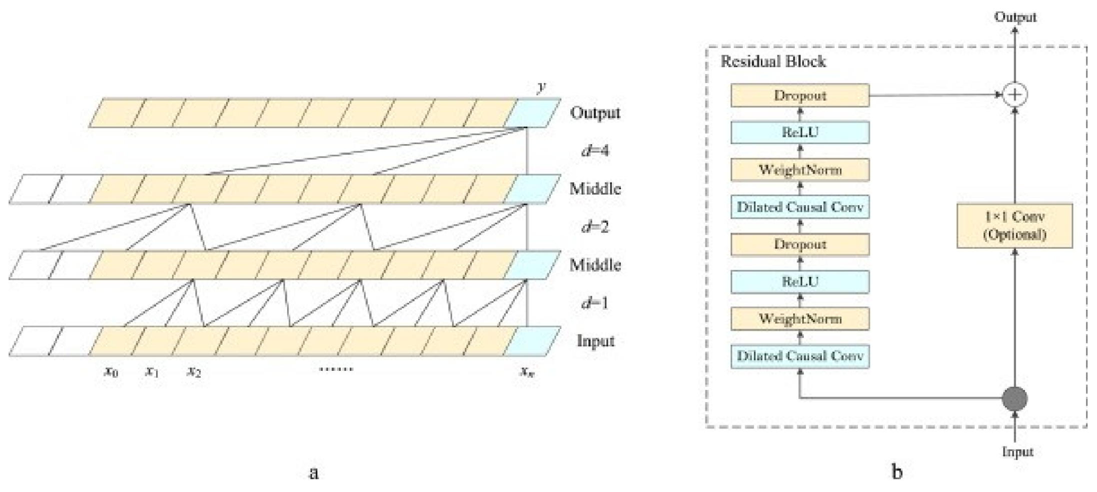


Figure 3.6: (a)A causal dilated convolution;(b)TCN residual block.Dilated convolutions [Zhu et al., 2020]

Dilated convolutions

Dilated convolutions enable TCNs to process time series with extended temporal dependencies efficiently. For a one-dimensional time series x and filter f, the dilated convolution operation F is defined as [Zhu et al., 2020]:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d\cdot i} \tag{3.1}$$

In this equation, k represents the filter size, d is the dilation factor, and s-d·i indicates the direction toward past values. When the dilation factor equals one, the operation reduces to regular convolution. As the dilation factor increases, the receptive field expands exponentially. TCNs can increase their receptive field through two primary methods: by increasing the dilation factor d or by using larger filter sizes k. This flexibility allows the network to adapt to various temporal scales and dependencies in the data.

Residual connections

Residual Networks (ResNets) [Lara-Benítez et al., 2020] are designed to address the performance degradation that occurs in deep CNNs when the number of layers becomes excessively large. In TCNs, the stacking of causal convolutions and dilated convolutions leads to progressively deeper networks. To prevent issues like gradient vanishing or attenuation during training, residual connections are introduced at the output layer of the TCN. These connections merge the input X directly with the output of the convolutional network, as expressed by the following equation:

$$o = Activation(x + F(x)) \tag{3.2}$$

Where $F(X)$ is the output of the convolutional layer and Activation(-) represents the activation function applied to the result. Residual connections allow the network to focus on refining the identity mapping rather than learning a complete transformation, which has been shown to improve the performance of very deep networks.

The residual block for TCN, shown in Figure 3.6, consists of two dilated causal convolutions, each followed by non-linearities. In this case, the Rectified Linear Unit (ReLU) function is used as the activation function. Additionally, weight normalization is applied to the convolutional filters, and an exclusion layer (dropout) is included after each dilated convolution to mitigate overfitting.

TCNs are particularly well-suited for sequence-to-sequence tasks due to their unique architecture and advantages:

- Parallelism: Unlike RNNs, where predictions at later time steps must wait for previous ones to complete, TCNs allow for parallel computation. Since the same convolutional filter is applied across all time steps in each layer, long input sequences can be processed simultaneously during both training and evaluation, significantly speeding up the process [Lara-Benítez et al., 2020].

- Flexible receptive field size: TCNs offer flexibility in adjusting the receptive field size, which can be customized by stacking more dilated convolutional layers, increasing dilation factors, or enlarging filter sizes. This adaptability gives TCNs greater control over model memory and makes them easily adjustable to different tasks and domains [Lara-Benítez et al., 2020].
- Low memory requirements: In contrast to LSTMs and GRUs, which require significant memory to store intermediate results from multiple gates, TCNs are more memory-efficient. The use of shared filters in each layer reduces memory usage, and the backpropagation path only depends on the depth of the network rather than the sequence length [Lara-Benítez et al., 2020].

3.9 Attention Mechanism

The attention mechanism emerged as a solution to address the limitations of early neural network architectures, particularly RNNs, which struggled to effectively process long sequences of data due to vanishing gradients and limited ability to capture long-range dependencies. Attention provides a dynamic framework, enabling models to selectively focus on the most relevant parts of the input sequence at each step, significantly improving their capacity to handle complex dependencies over extended sequences.

First introduced in the context of machine translation by Bahdanau et al. [Luong et al., 2015], the attention mechanism allowed models to identify which parts of the source sentence are most relevant to each word in the target sentence, leading to more accurate translations. This breakthrough laid the foundation for its widespread adoption across various domains in machine learning and natural language processing.

At its core, the attention mechanism computes a weighted sum of values, with the weights reflecting the importance or "attention" assigned to each element in the sequence. This can be formalized through the interaction of three key components:

- $Query(Q)$: Represents the element in the target sequence that requires attention.
- $Key(K)$: Represents the elements in the source sequence, which the query is compared against.
- $Value(V)$: Contains the information corresponding to each element in the source sequence, weighted by the attention scores.

For each query, the attention mechanism calculates a similarity score with each key, typically via a dot product. These scores are then normalized using the softmax function to produce a distribution of attention weights. These weights dictate how much influence each value should have on the output.

The attention computation is described by the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{3.3}$$

where d_k is the dimensionality of the key vectors.

Attention mechanisms can be classified into different types based on their scope and differentiability. One notable distinction is between global and local attention. Global attention attends to all elements of the input sequence when computing attention weights, providing a comprehensive overview of the sequence. Conversely, local attention restricts the focus to a subset of the input, reducing computational complexity and making it better suited for tasks involving very long sequences where global attention would be prohibitively expensive.

Another key distinction is between soft and hard attention. Soft attention calculates a weighted sum over all input elements and is differentiable, which makes it compatible with standard gradient-based optimization techniques. This differentiability has made soft attention the predominant form used in modern architectures. Hard attention, on the other hand, selects a single element from the input sequence, which reduces computational overhead but introduces non-differentiability. As a result, training hard attention models often requires specialized techniques like reinforcement learning. Hard attention is particularly useful in domains like visual processing, where selective focus on certain regions of an image is crucial.

3.10 Transformer

The success of attention mechanisms in improving sequence modeling tasks paved the way for the development of Transformers, a revolutionary architecture that relies entirely on attention mechanisms, eliminating the need for recurrent or convolutional layers. Introduced by Vaswani et al. [Vaswani et al., 2017], Transformers have become the foundation of many state-of-the-art models in NLP and beyond.

Transformers leverage the concept of self-attention, where the attention mechanism is applied to the same sequence for both queries, keys, and values. This allows the model to capture relationships between all elements of the sequence simultaneously, regardless of their distance from one another. The self-attention mechanism is the core building block of Transformers and is responsible for their ability to handle long-range dependencies effectively.

The Transformer architecture consists of two main components: the encoder and the decoder. Each component is composed of multiple layers of self-attention and feed-forward neural networks. The encoder processes the input sequence and generates a set of representations, while the decoder uses these representations to produce the output sequence. This architecture has proven to be highly effective for tasks such as machine translation, text summarization, and text generation.

One of the key innovations of Transformers is the use of multi-head attention, which allows the model to focus on different parts of the input sequence simultaneously. By computing multiple attention heads in parallel, the model can capture diverse patterns and relationships within the data, leading to richer and more nuanced representations.

Another important feature of Transformers is positional encoding, which is used to inject information about the position of each element in the sequence. Since Transformers do not rely on recurrence or convolution, they lack an inherent sense of sequence order. Positional encodings address this limitation by providing the model with information about the relative or absolute positions of elements in the sequence.

The success of Transformers has led to the development of numerous variants and extensions, such as BERT [Devlin et al., 2018], GPT [Radford and Narasimhan, 2018], and T5 [Raffel et al., 2019], which have achieved state-of-the-art performance on a wide range of NLP tasks. These models have demonstrated the versatility and scalability of the Transformer architecture, making it a cornerstone of modern machine learning.

3.11 Case-Based Reasoning (CBR)

Case-Based Reasoning (CBR) is an AI problem-solving approach that relies on the retrieval and reuse of past experiences or "cases" to address new problems. This methodology allows an AI system to make decisions or provide solutions by drawing upon its memory of previous similar situations and adapting them to the current context [Aamodt and Plaza, 1994]. has It been applied across various domains, including construction management [Hu et al., 2016], medical diagnosis [Pous et al., 2011], energy management [Pinto et al., 2019], and more.

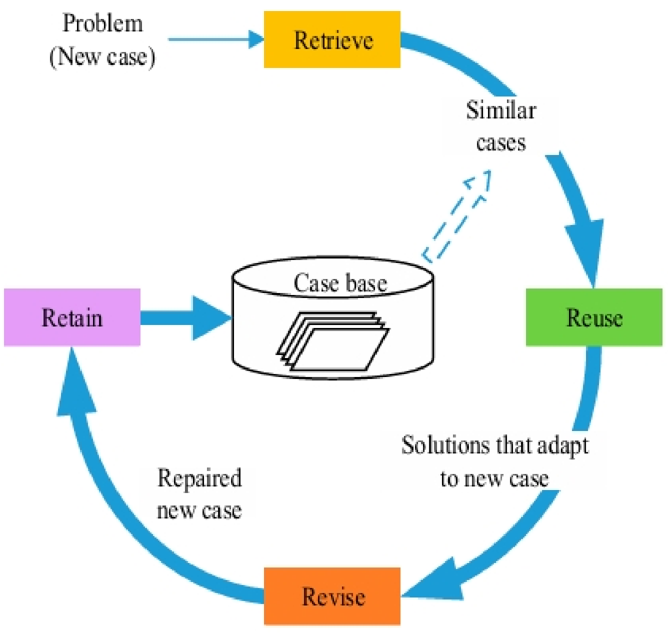


Figure 3.7: CBR cycle [Yan and Cheng, 2024]

The CBR process typically involves four main steps [Aamodt and Plaza, 1994] as show in the figure 3.7:

Retrieve: The core of the CBR process is the retrieval of previously encountered cases that are similar to the new problem at hand. This step involves searching the case base, which is the repository of past cases, to find the most relevant cases that can be adapted to solve the current problem.

Reuse: After the most relevant past cases have been retrieved, the next step in the CBR process is to reuse the knowledge and solutions from these cases to address the current problem.

The reuse phase involves adapting or modifying the solution from the retrieved case(s) to fit the specifics of the new problem. This step requires the extraction and transfer of relevant information, knowledge, and problem-solving strategies from the retrieved case(s) to the current situation.

Revise: After the reuse step, where the retrieved solution is adapted to fit the current problem, the CBR system enters the revise phase. In this step, the system evaluates the proposed solution and checks its validity and effectiveness in addressing the new problem.

Retain: The final step in the CBR process is the retain phase, where the system stores the new, successfully applied solution as a new case in its case base. This step is crucial for expanding the system's knowledge and improving its ability to solve similar problems in the future.

CBR is particularly useful in domains where explicit rules or algorithms may be difficult to define, and where experience and past cases play a crucial role in problem-solving. It allows AI systems to learn from their own experiences and gradually improve their performance over time [Richter and Weber, 2013].

By relying on past experiences, CBR can provide a complementary approach to data-driven techniques like deep learning. While deep learning excels at automatically extracting features and discovering complex patterns from large datasets, CBR can offer additional advantages in terms of reasoning, explanation, and the ability to handle novel situations that may not be well-represented in the training data.

The integration of CBR with deep learning has been an active area of research [Grace et al., 2016,Biswas et al., 2018,Hegdal and Kofod-Petersen, 2019], as it can lead to the development of hybrid systems that combine the strengths of both approaches. Such hybrid architectures can leverage the pattern recognition capabilities of deep learning while also incorporating the case-based reasoning capabilities to enhance the overall performance, robustness, and interpretability of the system.

3.12 Conclusion

In this chapter, we explored various machine learning and deep learning techniques, including CNN, RNN, LSTM, Bi-LSTM, GRU, TCN, attention mechanisms, and CBR. Each of these methods offers unique strengths for solving complex problems. Our discussion has laid the foundation for applying these techniques to address the challenges we face.

In the next chapter, we will focus on presenting our contributions, starting with detecting social engineering attacks.

Part II

Contribution

Chapter 4

Social engineering attack Detection

4.1 Introduction

In the ever-evolving landscape of cybersecurity, significant resources have been invested in developing robust technological defenses such as firewalls, encryption protocols, intrusion detection systems, and antivirus software. However, these advancements have not eliminated a critical vulnerability: the human element. As Heartfield and Loukas (2018) assert, people remain the weakest link in information security [Heartfield and Loukas, 2018]. This vulnerability stems from a fundamental aspect of human nature the tendency to place greater trust in other humans than in machines.

Social engineering attacks exploit this inherent trust, manipulating individuals into divulging confidential information or acting in ways that contravene security policies. These attacks pose a severe threat to both businesses and individuals, as they target human psychology rather than technical weaknesses. The efficacy of social engineering is particularly pronounced when human interaction is involved, rendering traditional software or hardware solutions inadequate without proper human training [Salahdine and Kaabouch, 2019].

The scale of this threat is starkly illustrated by the Verizon 2022 Data Breach Investigations Report, which found that an alarming 82% of analyzed breaches involved some form of human interaction [DBIR Report, 2022]. This statistic underscores the outsized role that human factors play in cybersecurity incidents. Cybercriminals employ a range of social engineering techniques, including phishing emails, vishing (voice phishing) calls, and pretexting messages. These methods are particularly favored when technical vulnerabilities in a system are scarce or well-defended [Aroyo et al., 2018].

As the internet has become an integral part of modern life, it has simultaneously created new security challenges. The human element in cybersecurity has become increasingly critical, with social engineering attacks evolving to exploit the complex interplay between human trust and technological systems.

In this chapter, we propose and evaluate machine learning approaches to detect social engineering attacks, focusing on well-established models such as SVM, Adaboost and XGBoost. Recognizing the inherent challenge of imbalanced datasets in cybersecurity (where legitimate interactions far outnumber malicious ones) we employ advanced oversampling techniques, specifically SMOTE-ENN (Synthetic Minority Over-sampling Technique with Edited Nearest Neighbors).

4.2 Architecture

This section describes our proposed machine learning-based approach for detecting social engineering attacks in more detail. Figure 6.1 provides a schematic overview of the methodology. Our study utilizes a real-world, imbalanced dataset of normal and malicious communications, originally compiled by [Lansley et al., 2020].

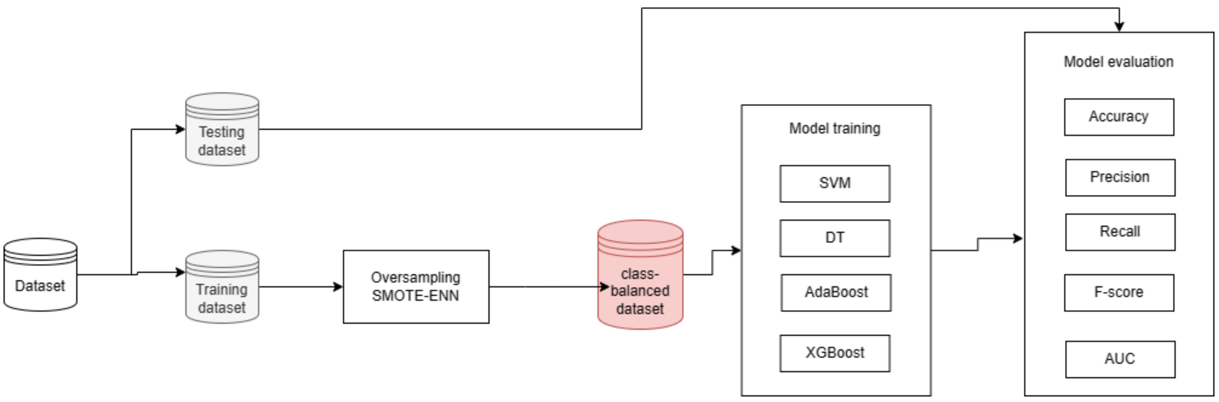


Figure 4.1: Architecture of the Social Engineering Attack Detection Model [Remmide et al., 2024b]

To mitigate the inherent class imbalance in the dataset, we employ the SMOTE-ENN. This sophisticated oversampling technique generates synthetic samples for the under-represented social engineering class while concurrently eliminating overlapping data points. This dual process serves to balance the dataset and enhance the model's capacity to detect social engineering attacks with improved accuracy.

We train multiple classifier models using the balanced dataset resulting from the SMOTE-ENN process. The models include SVM, AdaBoost, and XGBoost. These models represent a spectrum of traditional and ensemble machine learning techniques, each chosen for its suitability in handling complex classification tasks. The diversity in our model selection allows for a comprehensive evaluation of different approaches to social engineering detection.

Following the training phase, we evaluate the models on a separate hold-out test set. This evaluation assesses each model's detection capability for both normal and malicious messages, providing a robust measure of their performance in real-world scenarios.

Algorithm 2 presents the pseudocode for the key steps of our proposed approach, encapsulating the entire process from data preparation to model evaluation.

Algorithm 1 Pseudo code for the proposed approach to Social Engineering detection [Remmide et al., 2024b]

```
Input: Dataset
Output: Evaluation Measure (Accuracy, Precision, Recall, F-score and AUC )
 1: load(Dataset)
 2: Train, Test ← split(Dataset)
 3: Balance(Train)
 4: Initialize models
 5: For each fold in range(10)
 6: Evaluate Loss, Validation Loss
 7: Evaluate Accuracy, Validation Accuracy
 8: Evaluate Precision, Recall, F-score and AUC
 9: end for
10: return Average(Accuracy, Precision, Recall, F-score and AUC )
```

4.2.1 Data balancing

In our dataset, a significant class imbalance exists, with social engineering samples comprising only 10% of the total instances. This imbalance poses a challenge for machine learning models, potentially leading to biased predictions favoring the majority class. To address this issue, we employ the SMOTE-ENN algorithm [Batista et al., 2004], a sophisticated hybrid approach that combines oversampling of the minority class with undersampling of the majority class.

The choice of SMOTE-ENN is supported by its demonstrated superior performance in previous cybersecurity studies dealing with imbalanced datasets [Jain et al., 2022b, Tao et al., 2019]. Its effectiveness lies in its ability to not only balance class distributions but also improve decision boundaries by removing noisy and overlapping instances. This dual approach helps in creating a more representative and cleaner dataset, which is crucial for training robust machine learning models in the context of social engineering attack detection.

SMOTE-ENN operates in two main stages, each addressing different aspects of the class imbalance problem:

SMOTE-ENN operates through a two-stage process, each addressing different aspects of the class imbalance problem. The first stage, SMOTE (Synthetic Minority Oversampling Technique), generates synthetic samples for the under-represented social engineering class. It does this by selecting two or more similar instances based on k-nearest neighbors and creating new instances that are convex combinations of these selected instances. This approach balances the class distribution without simple duplication, thereby mitigating overfitting risks.

Following the SMOTE process, the second stage, ENN is applied to clean the resulting dataset. ENN examines each instance and its nearest neighbors, removing instances that are misclassified by their k-nearest neighbors. This cleaning step eliminates noisy or overlapping data points, which can significantly enhance the performance of the classifier by improving decision boundaries.

We implemented SMOTE-ENN with carefully tuned hyperparameters, as outlined in the following table 4.1:

Tableau 4.1: Summary of Hyperparameters used in SMOTE-ENN Implementation

Hyperparameter	**Value**
Oversampling Ratio	2:1
Number of Nearest Neighbors for SMOTE	5
ENN Cleaning Threshold (Tomek Links Removal)	3

4.2.2 classification model and parameters

In our study, we employ a diverse set of machine learning classifiers to detect instances of social engineering attacks. Our approach incorporates both traditional and ensemble learning techniques, specifically SVMs, AdaBoost, and XGBoost classifiers.

4.2.2.1 Support vector machines (SVM)

In our study, SVM was employed to classify instances of social engineering attacks versus legitimate interactions. We utilized the Radial Basis Function (RBF) kernel to handle non-linear relationships in the data, which allowed the model to map inputs into a higher-dimensional space where the attacks could be more easily distinguished from benign activities. The regularization parameter was carefully tuned to 1, striking a balance between maximizing the margin and controlling the model's complexity.

4.2.2.2 XGBoost

In our approach, XGBoost was utilized to predict whether an interaction constituted a social engineering attack or not. We optimized the tree method to "hist" for faster training and set the maximum tree depth to 3, ensuring that the model could capture important patterns without overcomplicating the decision boundaries. The regularized objective function helped control model complexity, allowing XGBoost to focus on meaningful interactions while avoiding overfitting to noise.

4.2.2.3 AdaBoost

For social engineering detection, we used AdaBoost to build a robust classifier by combining multiple weak decision tree learners. Each subsequent tree was trained to focus more on previously misclassified instances, allowing the model to adaptively improve its predictions over time. The base classifier used for AdaBoost was a decision tree, and the cluster size was set to 100, ensuring the model had enough flexibility to handle the complex and deceptive patterns typical of social engineering attacks. This iterative process allowed AdaBoost to progressively refine its understanding of social engineering threats.

We conducted a thorough hyperparameter tuning process to identify the optimal settings for each model. The hyperparameters were carefully selected using a combination of grid search and cross-validation to ensure optimal model performance. Grid search systematically explores a predefined range of hyperparameter values, while cross-validation evaluates the model's performance on different subsets of the data, ensuring robustness and generalizability. Table 4.2 summarizes the optimized hyperparameters used in our approach for social engineering attack detection.

Tableau 4.2: Optimized Hyperparameters for Social Engineering Attack Detection [Remmide et al., 2024b]

Models	Parameter	Value
AdaBoost	Base classifier	DT
	Cluster size	100
SVM	Kernel	RBF
	Regularization parameter	1
XgBoost	Tree method	hist
	maximum depth of the tree	3
	Cluster size	100

4.3 Evaluation and analysis

In this section, we describe our dataset, experimental setup, and evaluation methodology for social engineering attack detection. We evaluated several machine learning models, including SVM, XGBoost, Decision Trees, KNN, and AdaBoost, classifying messages as normal or social engineering attacks.

Performance was assessed using Accuracy (ACC), F1-score, and Area Under the Curve (AUC) to compare the effectiveness of our approach with existing methods. Additionally, one-way ANOVA was applied to statistically validate the performance differences, confirming the reliability and significance of our comparative results.

4.3.1 Analysis of Variance (ANOVA)

Analysis of Variance (ANOVA) is a statistical method used to compare the means of multiple groups by analyzing the variance within and between them. In this study, one-way ANOVA was employed to determine whether the observed differences in model performance (e.g., accuracy, F1-score, AUC) are statistically significant. This helps ensure that the results are not due to random chance but reflect true differences in the models' effectiveness.

The ANOVA test calculates an F-statistic, which is the ratio of between-group variance to within-group variance. A high F-statistic indicates that the differences between the models are significant. The results of the ANOVA test are further validated using a p-value, where a p-value less than a significance threshold (typically 0.05) confirms that the observed differences are statistically significant.

By applying ANOVA, we provide a rigorous statistical foundation for comparing the performance of our machine learning models, ensuring the reliability and validity of our findings.

4.3.2 Dataset overview

In this study, we utilized publicly available datasets compiled by previous researchers to train and evaluate our social engineering detection models. The primary dataset, referred to as the "compound dataset," was introduced by [Lansley et al., 2020] and consists of 1,051 entries. This dataset was created by merging several sources, including a standard dataset from [Bezuidenhout et al., 2010], which contributed 147 examples of social engineering attacks, and 600 tweets labeled as normal customer support interactions from Twitter.

Each entry in the dataset is represented by four key features:

- **Spelling:** Assesses the quality of spelling and the presence of slang, which attackers may use to appear less formal or bypass automated filters.
- **Links:** Evaluates the reputation of URLs within the text using the Web of Trust (WOT) API, identifying potentially malicious or suspicious links.
- **Intent:** Measures the presence of blacklisted words associated with social engineering attacks. This feature assigns a higher value to entries containing threatening or suspicious language, based on a predefined list of terms.
- **Attack Label:** A binary label indicating whether the text represents a social engineering attempt (1) or a normal interaction (0).

These features, extracted from the raw text as detailed by [Lansley et al., 2020], are designed to capture contextual cues that may indicate malicious intent. By converting these cues into numerical representations, they provide suitable input for machine learning algorithms, enabling them to identify patterns indicative of deceptive communications.

For model evaluation, we employed 10-fold cross-validation, where the dataset is divided into ten equal parts. In each iteration, one part is used for validation and the other nine for training. This process is repeated ten times, ensuring each entry is used for validation once and training nine times. This method provides a robust estimate of model performance by averaging results across all folds, reducing bias and variance in the evaluation.

4.3.3 Results

All experiments were conducted using Python 3.7 on a Windows 10 system equipped with an Intel Core i5 (10th Generation) processor, 16GB RAM, and an NVIDIA GTX 1060Ti GPU. The implementation utilized TensorFlow and Keras frameworks, along with CUDA toolkit version 5 for GPU acceleration.

Tableau 4.3: Comparative Performance Evaluation of Machine Learning Classifiers for Social Engineering Detection [Remmide et al., 2024b]

	Precision	**Recall**	**F-score**	**AUC**	**Accuracy**
AdaBoost	95,83%	30,71%	44,00%	85,33%	92,52%
SVM	83,17%	42,14%	53,80%	79,82%	92,91%
XgBoost	80,98%	37,50%	49,61%	91,67%	92,39%
KNN	26,27%	55,18%	34,91%	81,49%	77,40%
DT	82,67%	33,04%	45,16%	88,21%	92,11%
SMOT-ENN+SVM	**99,53%**	**96,28%**	**97,85%**	**99,86%**	**97,99%**

Our evaluation compared various machine learning classifiers on the social engineering detection dataset using standard performance metrics, as shown in Table 4.3 and Figure 4.2. The SMOTE-ENN oversampling technique demonstrated superior performance across all metrics, achieving over 96% for precision, recall, F-score, and AUC. This indicates its exceptional ability to accurately identify both social engineering attacks and legitimate text across diverse classification thresholds. The oversampled SVM model attained the highest overall performance, which can be attributed to the effective handling of class imbalance through oversampling and the kernel-based approach's particular suitability for capturing dataset-specific patterns.

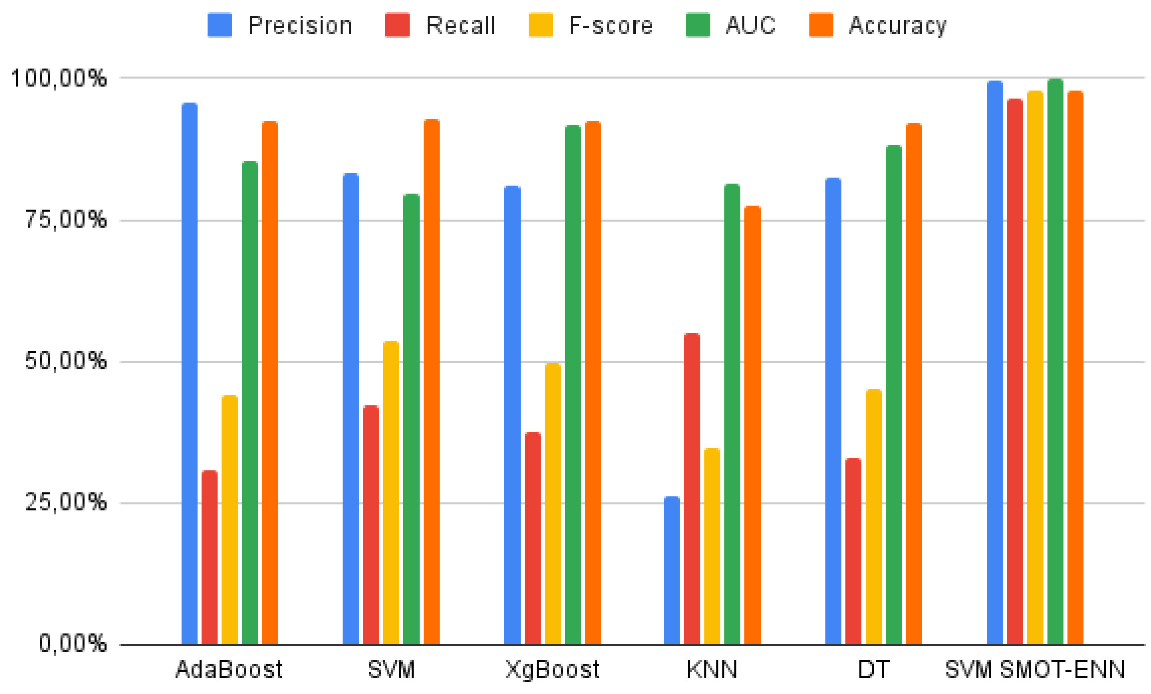


Figure 4.2: Evaluation of Machine Learning Algorithms for Recognizing Social Engineering Attacks [Remmide et al., 2024b]

AdaBoost demonstrated high precision (95.83%) but notably low recall (30.71%), suggesting a tendency to focus heavily on correctly classifying the majority class at the expense of missing a significant number of true social engineering attacks. Standard SVM and XGBoost exhibited similar performance profiles, balancing higher precision with moderate recall, though they may miss some attacks compared to the oversampled SVM. KNN, while achieving the highest recall among non-oversampled methods, suffered from significantly low precision (26.27%), resulting in a high false positive rate that limits its practical applicability.

Tableau 4.4: Evaluation of the Proposed Approach Compared to Benchmark Classifiers for Social Engineering Detection [Remmide et al., 2024b]

	F-score	AUC	Accuracy
MLP [Merton Lansley and Polatidis, 2020]	73,20%	68,86%	92,20%
Ensemble learning soft [Merton Lansley and Polatidis, 2020]	75,90%	72,22%	92,40%
Ensemble learning hard [Merton Lansley and Polatidis, 2020]	76,90%	72,20%	92,60%
SMOTE-ENN+SVM	97,85%	99,86%	97,99%

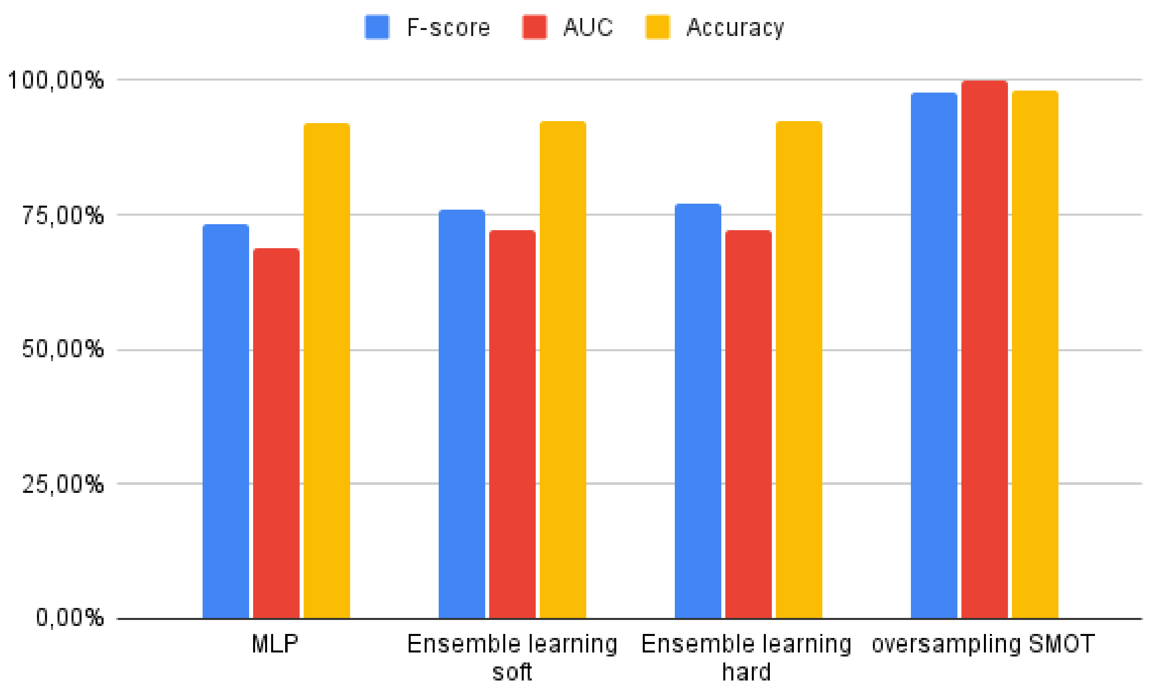


Figure 4.3: Performance Comparison of the Proposed Oversampling Method Against State-of-the-Art Classifiers for Social Engineering Detection [Remmide et al., 2024b]

Table 4.4 and Figure 4.3 present a comparison of our proposed SMOTE-ENN approach against benchmark classifiers from recent literature. Our method significantly outperformed existing techniques, achieving an F-score of 97.85% compared to approximately 75% for ensemble methods, an AUC of 99.86% versus roughly 72% for the best competing method, and an accuracy of 97.99% compared to 92.60% for ensemble learning with hard voting. Traditional approaches like MLP and ensemble learning, while effective for many classification tasks, demonstrated limitations in handling the unique challenges of social engineering detection.

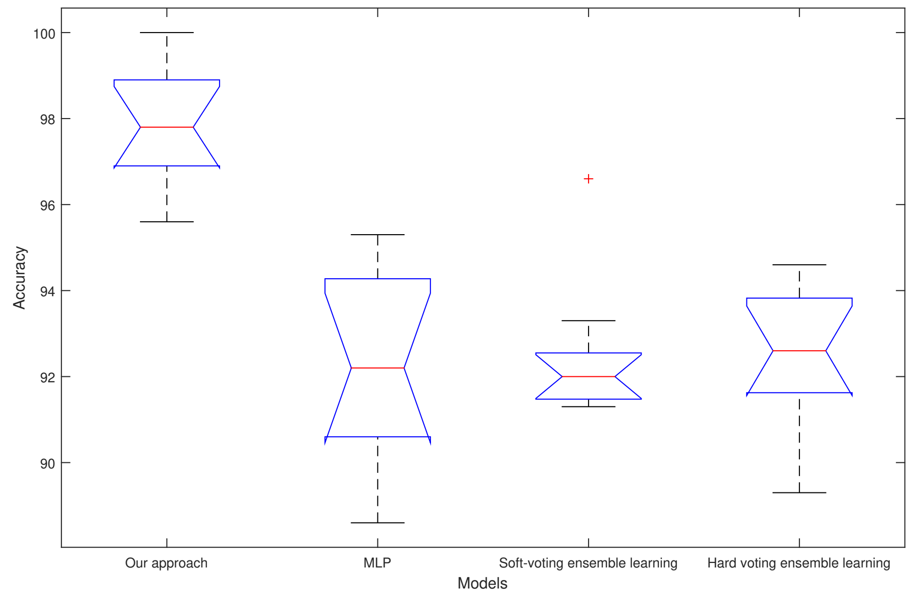


Figure 4.4: Accuracy Performance Comparison Across Various Social Engineering Detection Methods [Remmide et al., 2024b]

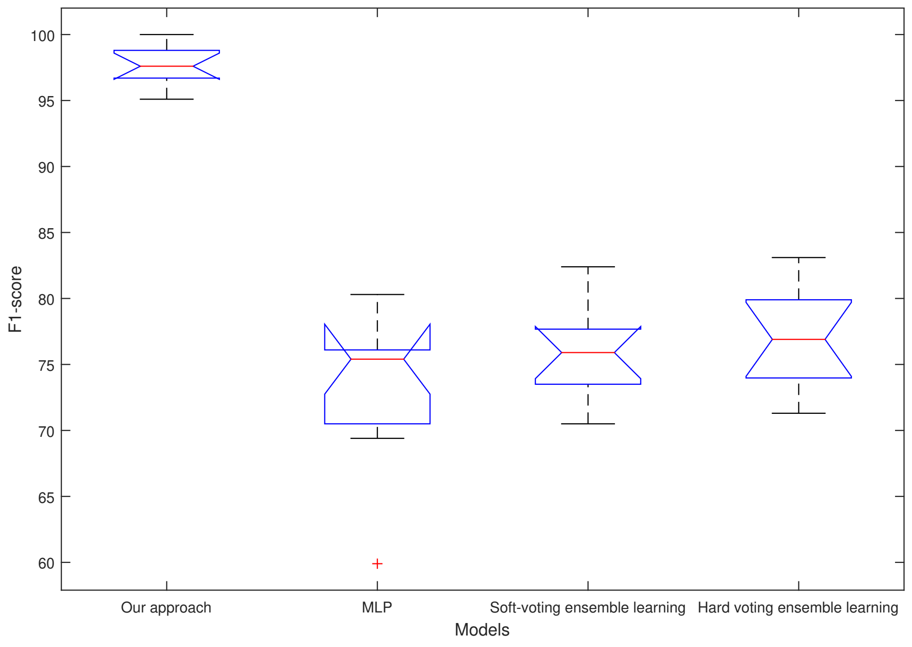


Figure 4.5: F1-score Performance Comparison Across Various Social Engineering Detection Methods [Remmide et al., 2024b]

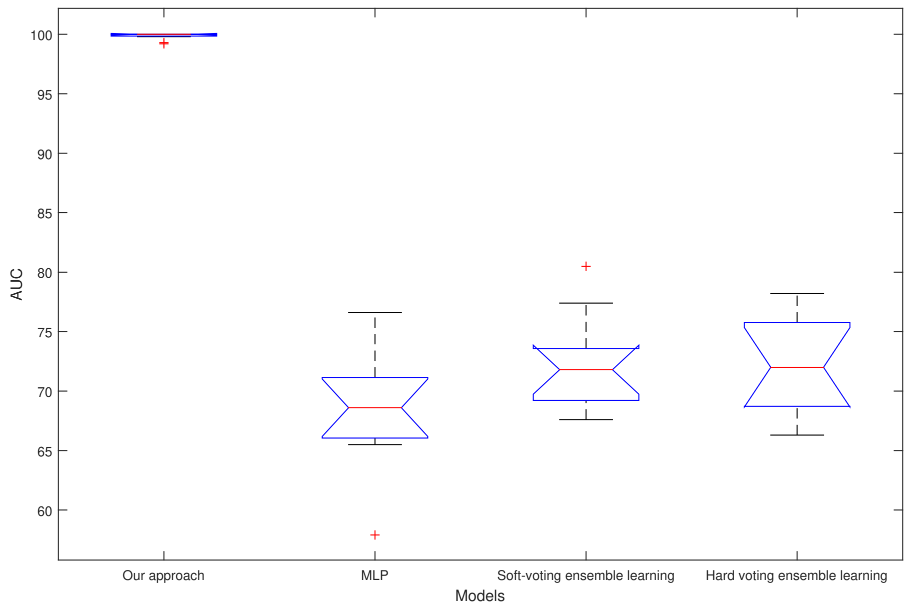


Figure 4.6: AUC Performance Comparison Across Various Social Engineering Detection Methods [Remmide et al., 2024b]

The performance metrics shown in Figures 4.4, 4.5, and 4.6 demonstrate the consistent superiority of our SMOTE-ENN approach across accuracy, F1-score, and AUC. Our method achieved near-perfect results, with a median accuracy significantly higher than competing models (p-value $= 4.88248 \times 10^{-10}$). Similarly, the F1-score comparison revealed an even greater statistical significance (p-value $= 9.04 \times 10^{-18}$), underscoring our method's ability to balance precision and recall effectively. The AUC results further validated our model's dominance, with a near-perfect median and the strongest statistical significance (p-value $= 1.125 \times 10^{-22}$). In comparison, traditional methods such as MLP and ensemble learning techniques showed lower median values and greater variability, highlighting their limitations in tackling the challenges of imbalanced datasets.

The superior performance of our method can be attributed to the effective handling of class imbalance through the SMOTE-ENN technique. By combining intelligent oversampling with ENN, our approach ensures that the minority class is well-represented while simultaneously eliminating noisy data points. This preprocessing step enables the model to form more accurate and robust decision boundaries, reducing bias towards the majority class. Additionally, the integration of strong base classifiers further enhances the model's ability to detect social engineering attacks with high sensitivity and precision.

These findings emphasize the critical importance of addressing data imbalance in social engineering detection systems. Traditional methods, while achieving reasonable performance on the majority class, struggle to effectively detect minority-class attacks, which are the primary concern in real-world applications. In contrast, our SMOTE-ENN approach delivers exceptional generalizability and robustness, making it a powerful solution for identifying and mitigating social engineering threats. This study underscores the need for specialized techniques tailored to the unique challenges of imbalanced datasets in cybersecurity.

4.4 Conclusion

This chapter presented a comprehensive approach to detecting social engineering attacks using machine learning techniques. Our methodology addressed several key challenges inherent in social engineering detection, particularly the significant class imbalance typically found in real-world datasets. Through extensive experimentation and analysis, we demonstrated the effectiveness of combining oversampling techniques, specifically SMOTE-ENN, with traditional machine learning classifiers.

The experimental results clearly showed the superiority of our proposed approach over existing methods. By achieving precision, recall, F-score, and AUC values exceeding 96%, our SMOTE-ENN enhanced classifiers significantly outperformed both baseline models and state-of-the-art techniques. The statistical analysis further confirmed these improvements, with p-values indicating highly significant advancements across all performance metrics. This success can be attributed to the effective handling of class imbalance, which enabled the creation of more representative decision boundaries and ultimately led to more robust detection capabilities.

Despite these promising results, our investigation highlighted a critical limitation in the field of social engineering detection: the scarcity of large, comprehensive datasets. This constraint poses a significant challenge to the development and validation of detection methods. Additionally, our analysis of attack patterns revealed that phishing remains the most prevalent form of social engineering attack, accounting for a substantial portion of all recorded incidents.

Given these findings and constraints, the focus of our subsequent research shifts to the specific domain of phishing detection, with particular emphasis on phishing URL identification. This transition is motivated by several factors: the abundance of available phishing URL datasets, the critical role URLs play in phishing attacks, and the potential for immediate practical applications in cybersecurity systems

Chapter 5

Phishing URL Detection

5.1 Introduction

In the ever-evolving landscape of cybersecurity threats, phishing remains one of the most pervasive and damaging forms of attack. Phishing is a sophisticated form of cybercrime that combines social engineering tactics with technical deception to illicitly acquire sensitive personal information, including financial account credentials and other confidential data [APWG, 2021b]. This malicious practice typically involves luring unsuspecting victims into clicking on deceptive links, often leading them to fraudulent websites designed to harvest their personal information. In more advanced scenarios, cybercriminals may employ technical subterfuge, surreptitiously installing malware on victims' devices to pilfer data directly.

The prevalence of phishing attacks has surged dramatically in recent years, posing a significant threat to both individuals and organizations worldwide. Its simplicity and low technical barriers make it highly accessible to attackers, contributing to its widespread use. According to the APWG, phishing attacks doubled between the first quarter of 2020 and the third quarter of 2021 [APWG, 2021b].

Particularly vulnerable to these attacks are sectors that handle sensitive user data and financial transactions. The APWG report highlights that software-as-a-service (SaaS) providers, webmail services, financial institutions, and payment processors are among the most frequently targeted entities [APWG, 2021b]. This targeting pattern reveals the attackers' strategic focus on sectors where successful breaches can yield the most valuable data.

This chapter delves into the detection of phishing URLs through Temporal Convolutional Networks (TCNs), a deep learning architecture particularly adept at sequence modeling tasks. By focusing on how TCNs can be employed to effectively identify phishing URLs, this chapter contributes to the development of stronger defenses against this ever-present cyber threat.

5.2 Architecture

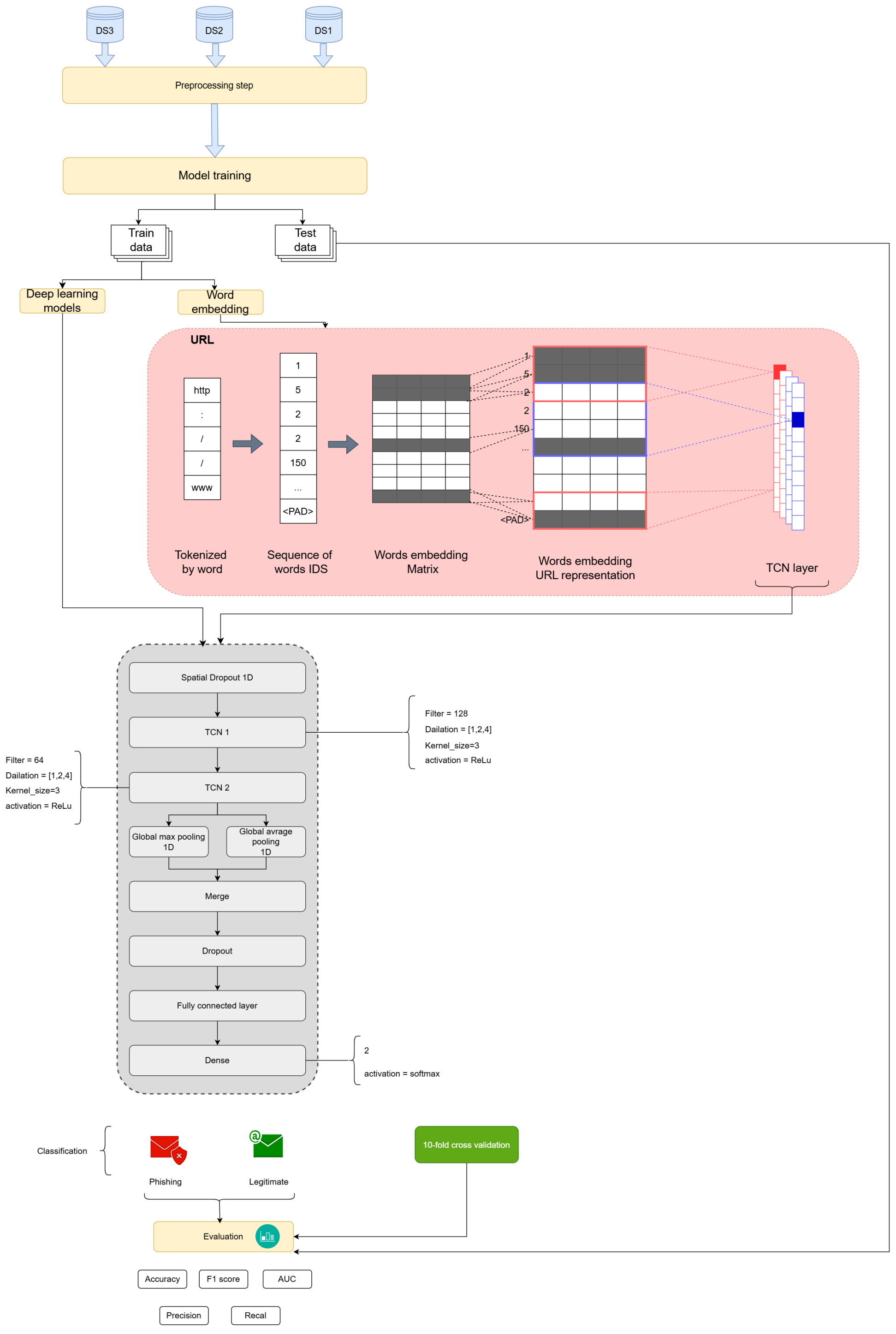


Figure 5.1: A high-level diagram of our the proposed phishing URL detection architecture

We propose the use of TCNs for phishing URL detection, as illustrated in Figure 5.1. The architecture processes URLs through data pre-processing and tokenization, converting URLs into word embeddings, followed by TCN layers that extract essential features for classification. The model ultimately classifies URLs as either legitimate or phishing. Finally, 10-fold cross-validation is used to evaluate performance based on accuracy, F1-score, precision, recall, and AUC.

TCNs are particularly effective for tasks like phishing URL detection due to their strengths in sequence modeling, which have proven valuable in NLP. Compared to RNNs, TCNs offer several advantages: they have a larger memory capacity for capturing long-term dependencies, achieve parallelism through convolutional layers for faster training, and allow for flexible receptive field sizes to enhance temporal awareness. Additionally, TCNs avoid the vanishing gradient problem commonly seen in RNNs, ensuring stable and efficient training.

5.3 Datasets

The effectiveness of a phishing URL detection system is heavily dependent on the quality and representativeness of the data used in its development. In our research, we selected three distinct datasets to provide a comprehensive evaluation of phishing URL detection techniques, considering various feature sets, sample sizes, and data distributions.

Dataset 1 :

The first dataset, as proposed by [Vrbančič et al., 2020], is available in two variants. The full version, referred to as Dataset-full, contains a total of 88,647 URL instances, of which 58,000 are legitimate and 30,647 are fraudulent. Each URL in this dataset is represented by a set of 111 characteristics that determine its legitimacy. A smaller version of this dataset, called Dataset-small, is also available. It contains 58,645 instances, with 27,998 legitimate URLs and 30,647 fraudulent URLs. Both versions of this dataset provide a rich set of features and a large number of samples, allowing for in-depth analysis of URL characteristics.

Dataset 2 :

The second dataset, sourced from PhishTank and Google[1]. It contains a total of 11,000 instances, evenly split between 5,500 legitimate URLs and 5,500 phishing URLs. This

[1]https://www.kaggle.com/datasets/sagarbanik/phishing-url-binary-datatset

dataset contains 14 unique features that differentiate phishing URLs from legitimate ones. This balanced dataset provides a concise set of features, making it suitable for developing and testing binary classification models.

Dataset 3 :

The third dataset, obtained from Kaggle[2], offers a large-scale collection of URL strings. It includes both the URL strings and their corresponding labels. The dataset contains a total of 549,346 URLs, of which 392,924 are legitimate and 156,422 are phishing URLs. This extensive dataset provides a real-world distribution of legitimate and phishing URLs, allowing for the development of models that can handle a large variety of URL patterns.

5.4 Preprocessing

Effective preprocessing is crucial for ensuring that our datasets are clean, well-structured, and optimally prepared for training models to detect phishing URLs. Each of our three datasets required specific preprocessing steps tailored to their unique characteristics and intended use in the study.

For Dataset 1 and Dataset 2, which contain structured features, we applied the following preprocessing steps:

1. **Normalization:** We scaled numerical features to ensure they were on a comparable scale, preventing any single feature from dominating the others due to differences in magnitude.

2. **Encoding of Categorical Features:** We employed one-hot encoding to transform categorical variables into a format suitable for machine learning algorithms. This process creates binary columns for each category, allowing the model to interpret categorical data effectively.

3. **Feature Scaling:** We applied standardization (z-score normalization) to ensure all features contributed equally to the model's learning process. This step involved subtracting the mean and dividing by the standard deviation for each feature.

4. **Handling Missing Values:** We addressed missing data by imputing values using statistical measures. Specifically, we used the mean of each feature to fill in missing values, ensuring data completeness without introducing significant bias.

[2]https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls

Dataset 3, which consists of raw URL strings, required a different approach focused on text processing:

1. **URL Parsing:** We decomposed each URL into its constituent parts (e.g., protocol, domain, path, query parameters) to extract meaningful features.
2. **Tokenization:** We split the URL strings into individual tokens, which could be words, subdomains, or other meaningful units within the URL structure.
3. **Text Normalization:** We converted all text to lowercase and removed special characters to reduce noise and ensure consistency across the dataset.
4. **GloVe Embeddings:** We utilized Global Vectors for Word Representation (GloVe) to convert our tokens into dense vector representations. This step allows the model to capture semantic relationships between different parts of the URLs.
5. **Sequence Padding:** To ensure uniform input length for our models, we applied padding to shorter sequences and truncated longer ones to a fixed length.
6. **Label Encoding:** We converted the binary classification labels (phishing/legitimate) into numerical format (0/1) for model compatibility.
7. **Data Cleaning:** We removed any corrupt or inconsistent data entries to ensure the quality of our dataset.

5.5 Data Balancing

To address the absorptive learning bias towards the dominant class (0) in our first database, we implemented a comprehensive data balancing strategy. Our approach employed two distinct techniques: subsampling and oversampling. By utilizing these methods, we aimed to create a more equitable representation of classes, thereby enhancing the reliability and accuracy of our subsequent analysis.

Subsampling Technique

We employed random undersampling to reduce the number of samples in the majority class (0). This technique helps mitigate the overfitting risk associated with the dominant class. Using the RandomUnderSampler from the imbalanced-learn library, we reduced the majority class from 58,645 to 30,647 samples, matching the count of the minority class (1).

Oversampling Technique

To complement the subsampling approach, we applied the Synthetic Minority Oversampling Technique (SMOTE) to increase the representation of the minority class (1). SMOTE generates synthetic examples in the feature space, avoiding exact replication and potential overfitting issues. Using the SMOTE implementation from imbalanced-learn, we increased the minority class from 30,647 to 58,645 samples, equaling the original count of the majority class.

5.6 Model

Our phishing detection system consists of two primary modules as shown in Figure 5.2. The word embedding learning module (using techniques such as Word2Vec and GloVe) performs the vector representation of words in URLs, while the detection module trains machine learning algorithms on these vector representations to classify URLs as phishing or legitimate. In this work, we adopt a TCN for its effectiveness and efficiency in sequence modeling tasks.

The architecture of our TCN-based model consists of several key components. The input layer accepts word embedding vectors of URLs, as raw URLs cannot be directly passed to the model. This is followed by two stacked TCN blocks, each with specific configurations. The first TCN block contains 128 filters, while the second uses 64 filters. Both blocks employ a kernel size of 3 and use expansion factors (dilation rates) of 1, 2, and 4 to capture different scales of temporal patterns.

After the TCN blocks, the model applies two parallel global pooling operations - max pooling and average pooling to the final sequence output. This dual pooling strategy helps capture different aspects of the sequence features. The results from both pooling operations are then concatenated and passed through a dense layer of 16 neurons, serving as a bottleneck for feature compression. Finally, the output layer consists of 2 neurons with a softmax activation function for binary classification (phishing vs. legitimate).

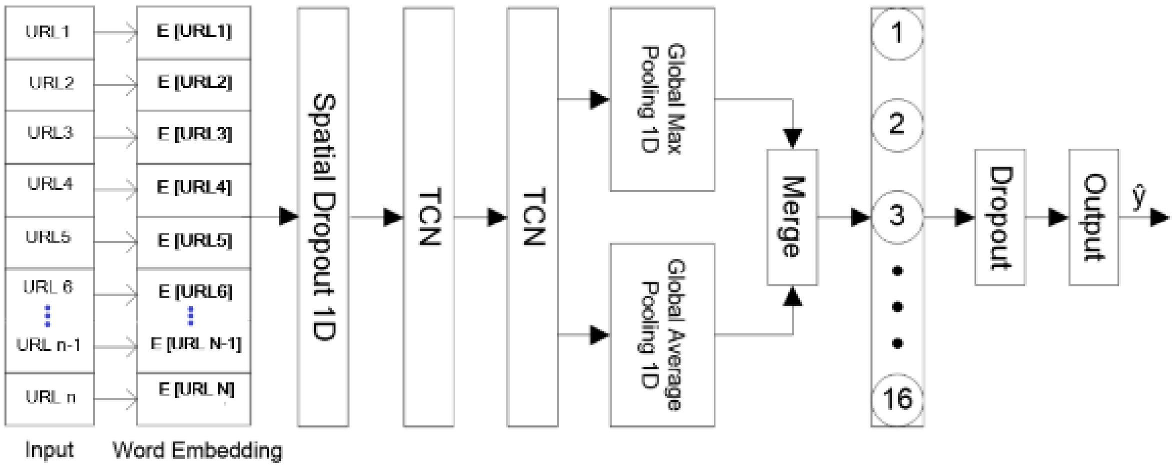


Figure 5.2: Our propose phishing URL detection model [Remmide et al., 2022b]

5.7 Experiments

All models were implemented using TensorFlow [Abadi et al., 2016] and Keras. We compared our TCN models with baseline classifiers such as KNN and Logistic Regression. we applied 10-fold cross-validation. Table 5.1 summarizes the performance of our TCN model across all datasets and configurations.

Tableau 5.1: Comparison of the proposed model in the three datasets [Remmide et al., 2022b]

Dataset	Model	Accuracy	Precision	Recall	F-1 score
2	**TCN**	**99,96%**	**99%**	**99%**	**99%**
1 (dataset_small)	TCN	98,76%	98%	98%	98%
1 (dataset_full)	TCN	96%	97%	96%	96%
1 (dataset_full) with undersampling	TCN	96%	96%	96%	96%
1 (dataset_full) with oversampling	TCN	98%	97%	97%	97%
3	Word2vec embedding with TCN	98,95%	98%	98%	98%
3	GloVe embedding with TCN	97,76%	98%	98%	98%

For Dataset 2, the TCN model performs exceptionally well, achieving an accuracy of 99.96%, with precision, recall, and F1 score all at 99%. This suggests that the model handles this dataset with near-perfect accuracy and minimal classification errors.

In Dataset 1 (small subset), the TCN model achieves a slightly lower accuracy of 98.76%, while still maintaining high precision, recall, and F1 score at 98%. This demonstrates that even with reduced data, the TCN model maintains strong performance.

For Dataset 1 (full), the model's performance decreases slightly with an accuracy of 96%, and precision, recall, and F1 scores all around 96-97%. This drop suggests that the larger dataset presents more challenges, potentially due to complexity or class imbalance.

When undersampling is applied to Dataset 1 (full), the performance remains consistent, with accuracy staying at 96%. This indicates that undersampling did not significantly impact model performance, possibly because of the inherent complexity of the data. However, when oversampling is applied, there is a noticeable improvement in accuracy, rising to 98%, with corresponding increases in precision, recall, and F1 score to 97%. This suggests that oversampling helps the model by addressing class imbalance more effectively than undersampling.

In Dataset 3, when Word2vec embeddings are used with the TCN, the model achieves a high accuracy of 98.95%, with other metrics at 98%. This demonstrates that the model benefits from the Word2vec embeddings, leading to robust performance. With GloVe embeddings, the accuracy drops slightly to 97.76%, although precision, recall, and F1 scores remain high at 98%. This indicates that while both embedding methods perform well, Word2vec offers a slight advantage.

In summary, the TCN model performs consistently well across different datasets, with oversampling showing notable improvements in handling data imbalance, and Word2vec embeddings offering the best results in dataset 3. The overall findings highlight the effectiveness of TCN, particularly when paired with techniques like oversampling and optimized embeddings.

Tables 5.2 and 5.3 compare our TCN model against baseline approaches for Datasets 1, 2, and 3.

Tableau 5.2: Performance Comparison of the Proposed Model and Baseline Models on the First and Second Datasets [Remmide et al., 2022b]

Dataset	Model	Accuracy	Precision	Recall	F-measure
D2	KNN	94.25%	94.2%	94%	94.10%
	TCN	99.96%	99%	99%	99%
D1 small	KNN	85.64%	85.4%	87.2%	86.31%
	TCN	98,76%	98%	98%	98%

Tableau 5.3: Performance Comparison of the Proposed Model and Baseline Models on the third Datasets [Remmide et al., 2022b]

Method	Accuracy	Precision	Recall	F1-score
KNN	91%	90%	86%	88%
LR + CountVectorizer	96%	93%	96%	95%
LR + RegexpTokenizer	96%	94%	96%	96%
Word2Vec + TCN	**98,95%**	**98%**	**98%**	**98%**
GloVe + TCN	97,76%	98%	98%	98%

In Dataset 2 (D2), the TCN significantly outperforms the KNN model, achieving an accuracy of 99.96% compared to KNN's 94.25%. Similarly, TCN achieves uniformly high precision, recall, and F-measure (99%), showcasing its superiority over KNN, which records slightly lower metrics (around 94%). For the smaller subset of Dataset 1 (D1 small), TCN again demonstrates a strong advantage with 98.76% accuracy, outperforming KNN's 85.64%. The high and consistent performance of TCN across both datasets highlights its robustness and effectiveness in comparison to the traditional KNN model.

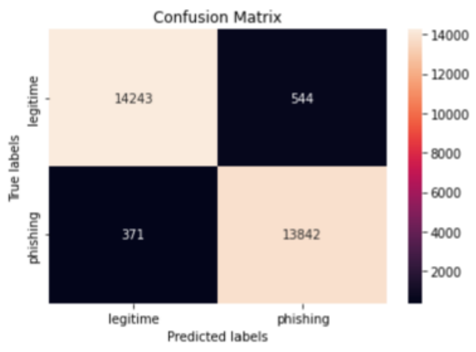
In the third dataset, the TCN model paired with Word2Vec embeddings achieves the highest performance, with an accuracy of 98.95% and precision, recall, and F1 scores all at 98%. This is superior to both KNN, which reaches only 91% accuracy, and LR combined with different feature extraction methods (CountVectorizer and RegexpTokenizer), which both achieve 96% accuracy. The TCN model paired with GloVe embeddings also performs well with an accuracy of 97.76%, but Word2Vec embeddings provide a slight edge. These results confirm that the TCN model, particularly when combined with advanced word embeddings like Word2Vec, outperforms traditional machine learning methods in handling complex datasets.

The confusion matrices in Figures 5.3a, 5.3b, and 5.3c illustrate a significant reduction in both false negatives and false positives across all experiments. This improvement is particularly notable in the second dataset, highlighting the effectiveness of the proposed model in enhancing classification accuracy and minimizing errors.

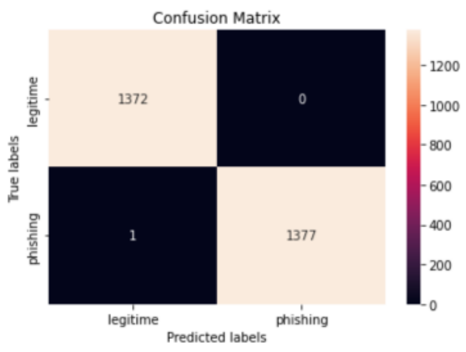
Tableau 5.4: Comparison of the Proposed Model and State-of-the-Art Methods [Remmide et al., 2022b]

Refrence	**Accuracy**	**Precision**	**Recall**	**F1-score**
[Abutair et al., 2019]	95%	-	-	96%
[Aljofey et al., 2020]	95.80%	96%	94%	95%
[Wang et al., 2019]	95.6%	97.33%	93.78%	95%
[Liang et al., 2019]	-	95.97%	95.96%	95.96%
[Liang et al., 2022]	99.27%	97.42%	95.44%	96.38%
Our model	**98,95%**	**98%**	**98%**	**98%**

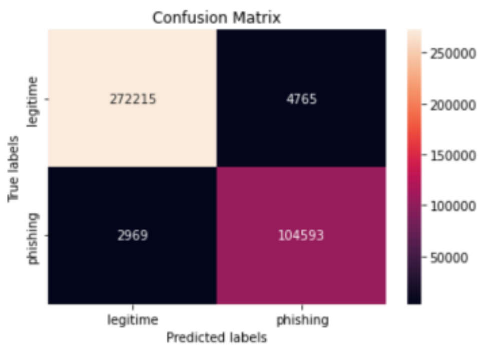
The table 5.4 compares the performance of the proposed model (Word2Vec + TCN) against several state-of-the-art models from prior studies. The proposed model achieves the highest overall performance, with an accuracy of 98.95%, and precision, recall, and F1-score all at 98%. This surpasses the accuracy of most of the compared models, such as the works by [Abutair et al., 2019] at 95%, [Aljofey et al., 2020] at 95.80%, and [Wang et al., 2019] at 95.6%, which also show slightly lower recall and precision values. While [Liang et al., 2019, Liang et al., 2022] achieved a higher accuracy at 99.27%, the precision and recall metrics 97.42% and 95.44%, respectively are slightly lower compared to the proposed model. Overall, this highlights that the proposed model delivers the most balanced and



(a) Confusion Matrix Displaying Results from the Test Set of Dataset 1.



(b) Confusion Matrix Displaying Results from the Test Set of Dataset 2.



(c) Confusion Matrix Displaying Results from the Test Set of Dataset 3.

Figure 5.3: Confusion Matrix of our TCN model Displaying Results from the 3 [Remmide et al., 2022b]

robust performance across all evaluation metrics, excelling in both classification accuracy and consistency.

5.8 Conclusion

In this chapter, we introduced and evaluated a TCN combined with word embeddings, a deep learning-based approach for phishing URL detection. Our experimental results demonstrated that the proposed model outperforms baseline models, achieving 98% in accuracy, precision, recall, and F1-score. Moving forward, our goal is to enhance the model's effectiveness against evolving phishing techniques and to validate its performance in real-world environments. In the next chapter, we will shift focus to detecting phishing emails by analyzing their content, further expanding the scope of our phishing detection efforts.

Chapter 6

Deep Learning and Case-Based Reasoning for Phishing Email Detection

6.1 Introduction

In the digital age, the Internet and technological advancements have become integral to both professional and personal life. However, this increased reliance on technology has led to a surge in cybercriminal activity. The Deep Instinct report highlighted a staggering 653% increase in malicious activity in July 2020 compared to the previous year [Cyber Threat, 2020]. Moreover, the U.S. Healthcare Cybersecurity Market 2020 report revealed that over 90% of healthcare organizations experienced at least one cybersecurity breach in the past three years [1]. These statistics underscore the critical importance of cybersecurity in today's interconnected world.

Despite the implementation of advanced security technologies, attackers continue to find sophisticated methods to exploit the weakest link in any system: the human element. Verizon's 2021 DBIR indicates that a staggering 85% of breaches are attributable to human factors [DBIR Report, 2021]. This vulnerability is often exploited through social engineering techniques, with phishing attacks emerging as the most prevalent form of cyberthreat.

Phishing attacks, which typically involve deceiving victims into divulging confidential information or performing harmful actions, have seen a dramatic rise in recent years. The APWG reported a tripling of phishing attacks since the beginning of 2020, with the financial sector remaining the primary target [APWG, 2021a]. December 2021 marked a historic high in APWG's reporting, with 316,747 recorded phishing attacks [APWG, 2021a].

[1] https://www.gminsights.com/industry-analysis/healthcare-cybersecurity-market

While traditional phishing attacks predominantly utilized email as the primary vector, cybercriminals have expanded their tactics to include other communication channels. Smishing (SMS phishing) targets users through text messages, while vishing exploits Voice over IP (VoIP) communications [Gupta et al., 2015]. Social media platforms have also become breeding grounds for phishing activity, with the APWG Q4 2022 report indicating that approximately 10.5% of all phishing attacks targeted social media users [APWG, 2022].

The dynamic nature of phishing attacks presents a significant challenge, as attackers continuously evolve their techniques to evade detection. This adaptability underscores the need for more robust and effective anti-phishing solutions [Basit et al., 2021, Bountakas and Xenakis, 2023]. In response to this challenge, researchers have begun exploring hybrid approaches that combine the strengths of multiple techniques to enhance phishing detection capabilities.

One promising avenue of research is the integration of CBR with deep learning techniques for phishing email detection. Case-based reasoning, a problem-solving paradigm that utilizes past experiences to address new problems, offers the potential to complement the pattern recognition capabilities of deep learning models. By leveraging historical phishing cases and the adaptive learning capabilities of deep neural networks, a hybrid CBR-deep learning approach could potentially offer improved detection accuracy and resilience against evolving phishing tactics.

This chapter explores the synergistic potential of combining case-based reasoning and deep learning for content-based phishing email detection.

6.2 Architecture

This chapter presents phishing email detection as a classification problem, proposing a novel hybrid approach that integrates deep learning with CBR. Our methodology leverages deep learning techniques to construct optimal case representations, which are subsequently utilized within the CBR cycle to enhance classification accuracy.

We introduce Deep Learning-augmented Case-Based Reasoning (DL-CBR), a comprehensive framework that encompasses five interconnected stages within a single cycle: Representation, Retrieve, Reuse, Retain, and Revise (5R). Figure 6.1 illustrates the architectural overview of our model, delineating these five stages and their intricate interactions.

Prior to deploying the model for phishing email detection, two crucial initialization steps must be completed. First, the deep learning feature extraction model must undergo thorough training using a designated training dataset. Second, the system's case base must be populated with a diverse set of carefully curated cases, establishing the foundational knowledge for the CBR component. This dual initialization process ensures that both the deep learning and CBR components are adequately prepared for effective phishing detection.

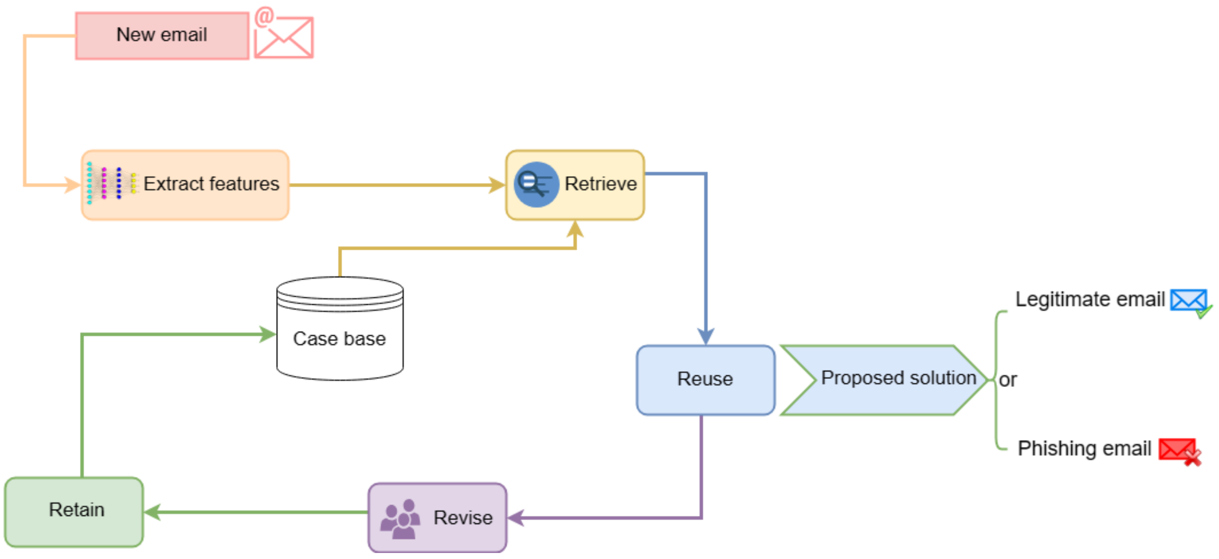


Figure 6.1: Overview of the proposed Deep Learning-augmented Case-Based Reasoning (DL-CBR) system [Remmide et al., 2024c]

6.2.1 Representation (feature extraction)

CBR is a problem-solving paradigm that utilizes past experiences to address new challenges. In our context of phishing email detection, each new email represents a case that can be modeled in various formats, such as feature vectors, structured data, or textual representations [Bergmann et al., 2005]. This research adopts a feature vector representation, extracted using a neural network. The process starts with feeding the system the incoming emails, which are preprocessed to isolate the body text. After removing noise and irrelevant elements such as HTML tags, hyperlinks, encoded characters, punctuation, and stop words the emails are tokenized at the word level.

The neural network architecture consists of a TCN, a Bi-LSTM layer, and a final dense layer. However, since we are not training a conventional classifier, standard learning methods are not applicable. Instead, we employ a more advanced technique: the N-pair multi-class loss function. Figure 6.2 illustrates the overall architecture of our deep learning-based feature extraction, and each component is detailed in the following subsections.

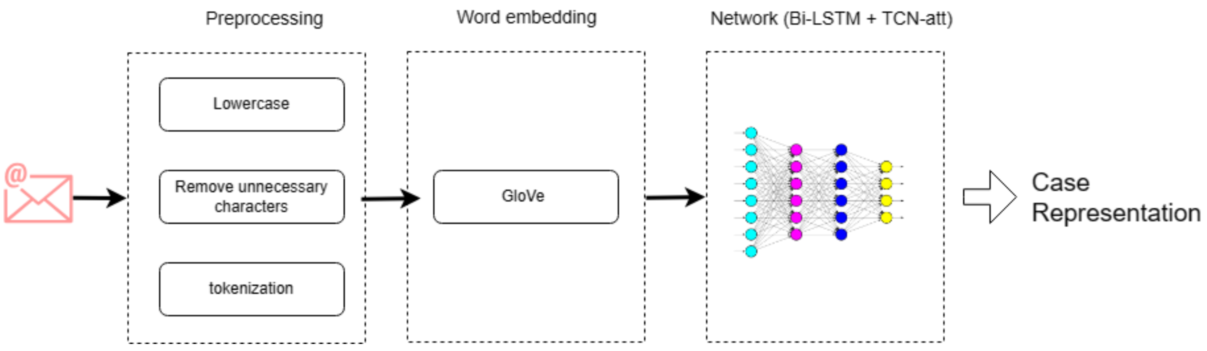


Figure 6.2: Overview of the feature extraction process steps [Remmide et al., 2024c]

6.2.1.1 Data preprocessing

Preprocessing is a key step in NLP, particularly when working with email data. It transforms raw text into a cleaner and more structured form, ready for analysis. This process involves several important tasks aimed at reducing noise and ensuring consistency across the data.

The first step is case normalization, where all text is converted to lowercase. This prevents the same word from being treated differently due to capitalization. Next, common stop words like "a," "the," and "is" are removed because they occur frequently but do not add much meaning to the text.

After that, we remove unnecessary symbols and punctuation (e.g., commas, periods, brackets) that do not typically contribute to the meaning of the content. For email-specific elements, such as email addresses, URLs, or signatures, special handling is applied they are removed.

Finally, tokenization is performed. This step breaks the text into smaller units (words) making it easier for later steps. The pseudocode for this preprocessing sequence is presented in Algorithm 2.

Algorithm 2 Algorithm of data preprocessing [Remmide et al., 2024c]

```
Input: E-mail messages
Output: list of words
1: list ← []
2: For each E-mail message E
3: LowerCase(E)
4: Remove(punctuation, stopword, numbers, whitespace characters from E)
5: list.append(Split(E))
6: end for
7: return list
```

6.2.1.2 Word embedding

Word embedding is a crucial technique in NLP that transforms raw textual data into vector representations suitable for machine learning and deep learning algorithms. This process maps words to dense vectors of real values, preserving semantic relationships and contextual information. For instance, words with similar meanings, such as "king" and "queen," are positioned closely in the embedding space, while semantically disparate words like "joyful" and "miserable" are situated far apart.

To enhance generalization, especially when working with limited training data, word embeddings are typically pre-trained on extensive, unannotated corpora. Several prominent embedding techniques exist, including Word2Vec, GloVe [Pennington et al., 2014], and FastText. This study employs GloVe embeddings, specifically those trained by Pennington et al. [Pennington et al., 2014] on a corpus of 27 billion tokens from Twitter tweets.

The embedding process for a given text input, such as an email, is illustrated in Figure 6.3. The embedding layer takes a sequence of words, denoted as $x_1, ..., x_n$, where $n \leq 100$, meaning we consider a maximum of 100 words per email. Each word x_i is converted into a 100-dimensional GloVe vector, which captures its meaning based on the context in which it appears.

Formally, let $E : V \to \mathbb{R}^d$ be the embedding function that maps words from the vocabulary V to d-dimensional vectors (in this case, $d = 100$). The email can then be represented as a matrix $X \in \mathbb{R}^{n \times d}$, where:

$$X = \begin{bmatrix} E(x_1) \\ E(x_2) \\ \vdots \\ E(x_n) \end{bmatrix}$$

This representation preserves both the semantic meaning of individual words and their sequential order within the email. The output of the embedding layer for each email is a two-dimensional array of shape $(100, 100)$, where each row corresponds to a word vector, and each column represents one of the 100 dimensions of the GloVe vectors.

Using this embedding method, we turn the raw email text into a meaningful representation that captures relationships between words, which helps improve classification task.

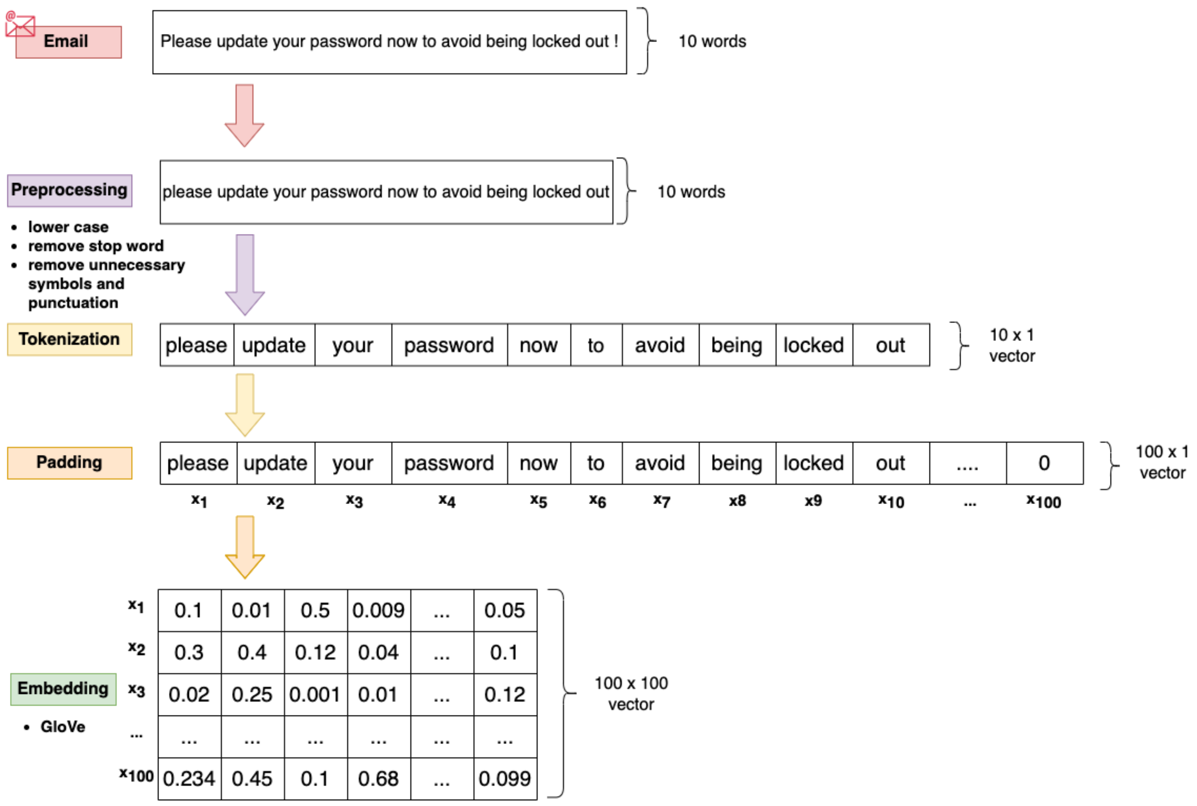

Email
Please update your password now to avoid being locked out !
10 words
Preprocessing
• lower case
• remove stop word
• remove unnecessary symbols and punctuation
please update your password now to avoid being locked out
10 words
Tokenization
please | update | your | password | now | to | avoid | being | locked | out
10 x 1 vector
Padding
please | update | your | password | now | to | avoid | being | locked | out | .... | 0
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 ... x100
100 x 1 vector
Embedding
• GloVe
x1: 0.1 | 0.01 | 0.5 | 0.009 | ... | 0.05
x2: 0.3 | 0.4 | 0.12 | 0.04 | ... | 0.1
x3: 0.02 | 0.25 | 0.001 | 0.01 | ... | 0.12
...: ... | ... | ... | ... | ... | ...
x100: 0.234 | 0.45 | 0.1 | 0.68 | ... | 0.099
100 x 100 vector

Figure 6.3: Word embedding process for a single email [Remmide et al., 2024c]

6.2.1.3 Network (Bi-LSTM+TCN model with attention)

In this study, we introduce a hybrid model that combines Bi-LSTM networks and TCN with an attention mechanism. This architecture is designed to capture both sequential and contextual dependencies from the email data, while focusing on the most relevant information for case representation. Figure 6.4 provides an overview of the model's architecture.

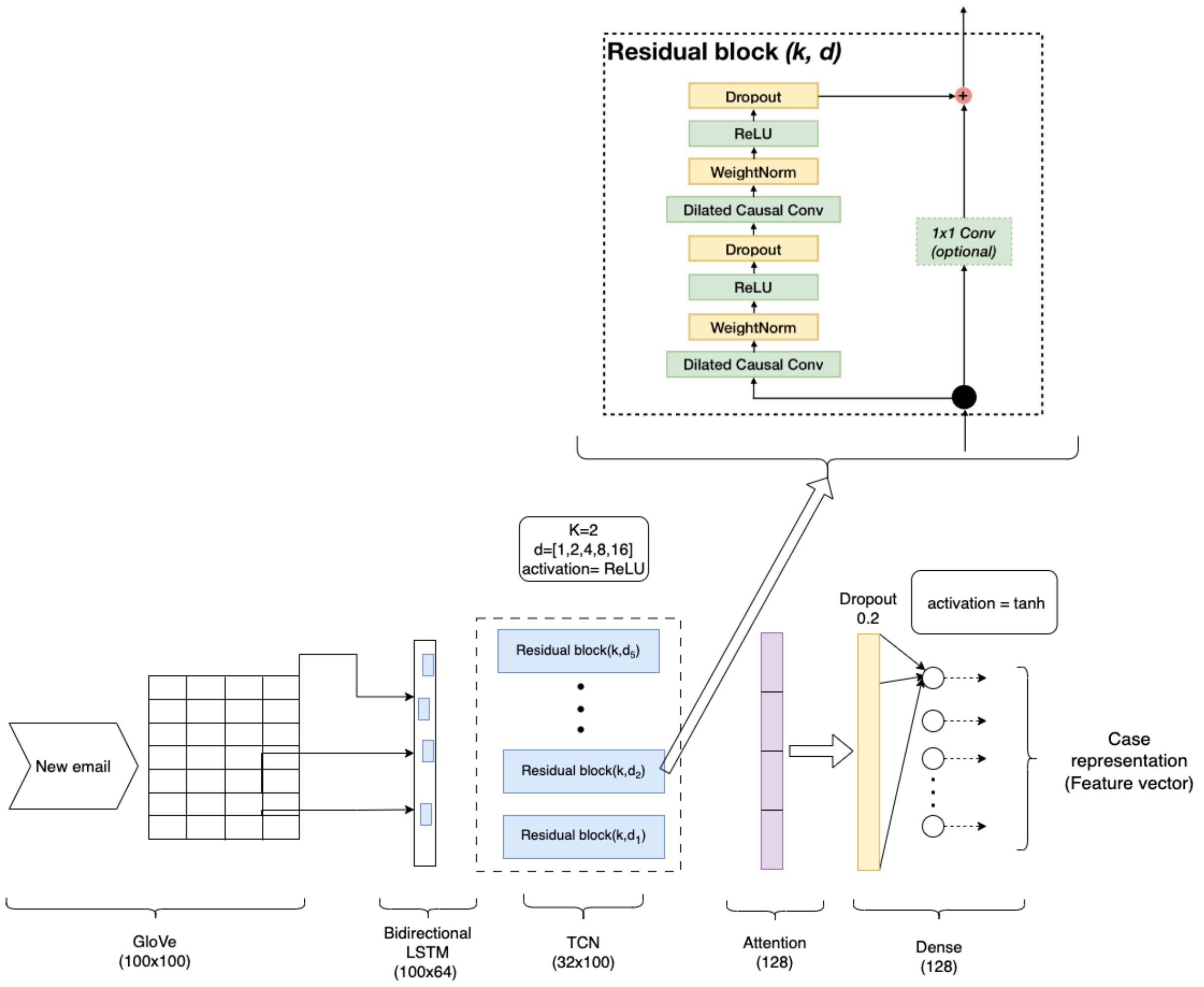


Figure 6.4: Architecture of the proposed deep learning model for feature extraction [Remmide et al., 2024c]

Bidirectional LSTM Layer The model begins with a Bi-LSTM layer [Hochreiter and Schmidhuber, 1997, Schuster and Paliwal, 1997], which extracts semantic features from the input sequence. Bi-LSTM networks consist of three gates: the forget gate, input gate, and output gate, denoted as f_t, i_t, and o_t, respectively. The forget gate determines which information to discard from the cell state, while the input gate selects which new information to store. The cell state is updated accordingly, and the output gate generates the final output.

The Bi-LSTM cell operates on three inputs: the hidden state h_{t-1}, the memory state c_{t-1}, and the current input x_t, representing the t-th word in the email. The cell's output is computed using the following equations:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{6.1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{6.2}$$

$$\tilde{c}_i = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{6.3}$$

$$c_i = f_i \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{6.4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{6.5}$$

$$h_t = o_t \otimes tanh(c_t) \tag{6.6}$$

Where W_j,U_j are weight matrices that represent learning parameters and b_j are the bias vectors for Bi-LSTM $j \in \{i, f, o, c\}$. The symbols $\sigma(\cdot)$ and $tanh(\cdot)$ denote the element-wise sigmoid and hyperbolic tangent functions, and $\otimes$ is the element-wise multiplication. The initial values of h_0 and c_0 are 0. h_t is the hidden state is the final output of the Bi-LSTM at time t.

Temporal Convolutional Network Layer The second layer is the TCN layer [Bai et al., 2018,Remmide et al., 2022b,Aouchiche et al., 2024], which performs deeper feature extraction. The TCN receives the final hidden state h_t from the Bi-LSTM and processes it using dilated convolutions to increase the receptive field. The TCN also incorporates residual connections to stabilize training and prevent gradient issues. The operation for a dilated convolution is defined as:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d\cdot i} \tag{6.7}$$

Where d represents the dilation factor, k is the size of the convolution kernel, and $s - d \cdot i$ determines the receptive field size.

Attention Mechanism The third layer is the attention mechanism [Bahdanau et al., 2014], which enhances the model's focus on key information while ignoring less relevant details. The attention mechanism computes a weight vector that assigns importance to each word in the sequence. The attention score is calculated as:

$$U = tanh(W_w X + b_w) \tag{6.8}$$

$$a = softmax(U) \tag{6.9}$$

$$c = Xa^T \tag{6.10}$$

Here, $W_w$and b_w are the weight and bias parameters of the attention layer, X represents the output from the previous layer, and a is the attention score that determines which parts of the input should be focused on.

Following the attention layer, we apply dropout (with a 0.2 rate) to prevent overfitting, followed by a dense layer with hyperbolic tangent activation. The dense layer outputs a vector that represents the case of the email, with the size of this layer determining the dimensionality of the case representation. Figure 6.5 illustrates the step-by-step processing of an input sequence through each layer of the model.

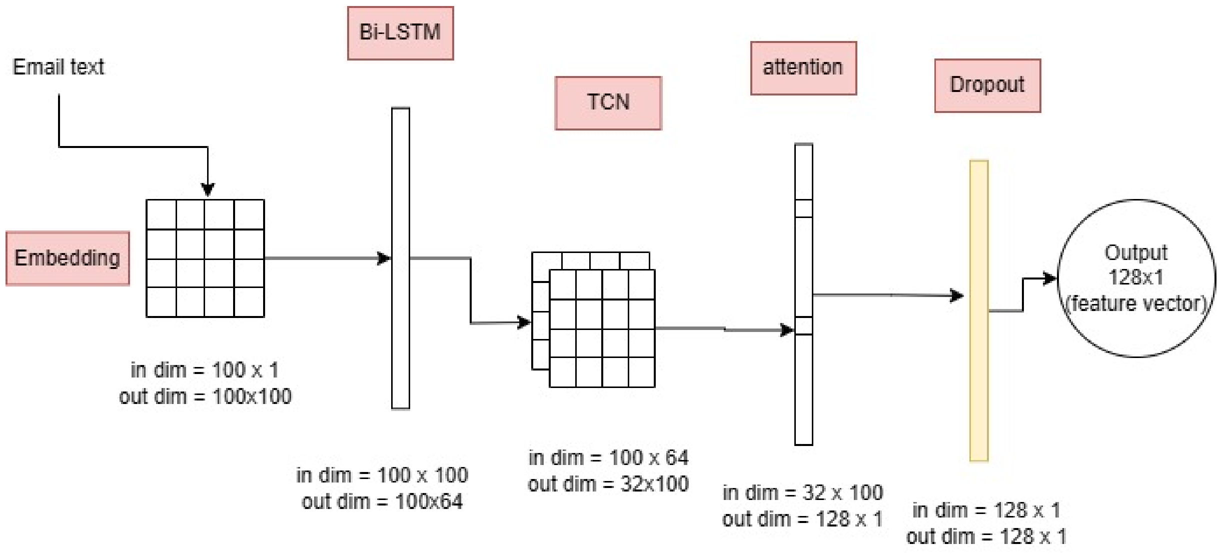


Figure 6.5: Diagram detailing inputs and outputs at each stage of the feature extraction deep learning model. [Remmide et al., 2024c]

6.2.1.4 Multi-class N-pair loss function

Since our goal is to generate effective representations of email cases, traditional training methods—based on calculating the loss between predicted and true outputs—are not applicable. Instead, we employ a deep metric learning approach known as the multi-class N-pair loss function [Sohn, 2016].

The multi-class N-pair loss function is designed to bring similar emails from the same class closer in the embedding space, while pushing apart emails from different classes. It extends the concept of triplet loss by simultaneously considering multiple negative examples rather than a single negative example for each update. This increases the efficiency and robustness of the training process.

Given N pairs of emails $\{(x_1, x_1^+), ..., (x_N, x_N^+)\}$ from N classes and feature vector f, the loss is formulated as follows (eq. 6.11) :

$$L_{N-pair-mc}(\{(x_i, x_i^+)\}_{i=1}^N; f) = \frac{1}{N}\sum_{i=1}^{N}\log\left(1 + \sum_{j\neq i}\exp(f(x_i)^\top f(x_j^+) - f(x_i)^\top f(x_i^+))\right) \tag{6.11}$$

Where N represents the total number of email pairs with matching labels, f_i is the embedding of the email x_i and f^+ is the embedding of its corresponding positive example x_i^+. The loss function works by minimizing the distance between positive email pairs while maximizing the distance between negative pairs (i.e., emails from different classes). This helps in creating a more discriminative and meaningful email representation for subsequent classification task.

6.2.2 Retrieve

The retrieve stage operates on email representations generated by the neural network in the feature extraction phase. These representations, along with their corresponding classifications, populate the case base in a high-dimensional embedding space.

For new email classification, the process follows two key steps. First, the neural network generates an embedding representation for the incoming email. Second, the system queries the case base to identify similar, previously classified cases using a k-NN approach.

The similarity between cases is quantified using Euclidean distance in the embedding space. For two cases P and Q, each with n features, the Euclidean distance is calculated as:

$$d(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{6.12}$$

where p_i represents the i-th feature of case P, q_i represents the i-th feature of case Q, and n is the total number of features.

The retrieval process involves computing similarities between the new case and all cases in the case base, ordering cases by decreasing similarity (increasing Euclidean distance), and selecting the top K most similar cases. In our DL-CBR implementation, we set K=3, retrieving the three most similar cases for subsequent processing.

6.2.3 Reuse

The reuse stage is critical for determining a solution to the new problem based on retrieved cases. This process can range from directly applying the solution of the most similar case to requiring sophisticated adaptation mechanisms, depending on the similarity between retrieved cases and the new problem.

In our approach, we implement a weighted voting mechanism using the K most similar retrieved cases. Each case contributes a vote towards either the 'phishing' or 'legitimate' classification based on its stored label. The final classification is determined by majority rule, where the class receiving the most votes becomes the predicted label for the new case.

However, our system incorporates an important safety mechanism: if all retrieved cases exhibit low similarity scores (falling below a predetermined threshold) to the new case, the DL-CBR system automatically classifies the email as phishing and flag it for humain validation. This conservative approach is based on the principle that if an email doesn't closely resemble any known legitimate emails in the case base, it warrants careful scrutiny as potentially malicious.

This default-to-phishing strategy reflects our system's design philosophy of prioritizing security over convenience. We deliberately optimize to minimize false negatives (undetected phishing attempts) at the potential cost of increasing false positives (legitimate emails incorrectly flagged as phishing). This trade-off acknowledges that the consequences of missing a phishing attack typically outweigh the inconvenience of manually verifying a falsely flagged legitimate email.

6.2.4 Revise

The revision stage is crucial for enabling continuous learning in our Case-Based Reasoning system. Through this stage, the system evolves and improves by learning from both successes and failures, with human experts playing a pivotal role in the verification and correction process.

Our implementation employs a hybrid human-in-the-loop approach. When the system encounters emails with similarity scores below a predefined confidence threshold, these cases are automatically flagged for expert review. Domain specialists then examine these flagged emails, leveraging their expertise to either validate the system's classification or provide corrective feedback. This expert-driven revision process operates through several key mechanisms:

- **Manual Validation:** Human reviewers examine flagged emails, confirming or overriding the system's initial classification.
- **Feedback Integration:** Cases where expert assessment differs from the system's prediction are incorporated into the case base with the corrected classification.

- **Proactive Review:** Domain experts can initiate reviews of any emails they suspect may have been misclassified, regardless of the system's confidence level.
- **Customization Capability:** The system supports adaptation to specific organizational requirements through expert feedback, allowing for fine-tuning of classification criteria.

This iterative feedback loop significantly enhances the robustness of our DL-CBR system. By continuously incorporating expert knowledge, the system becomes increasingly adept at handling ambiguous or borderline cases that initially fell below the confidence threshold. Importantly, this approach maintains a balance between automated efficiency and human expertise, ensuring both scalability and accuracy in email classification.

6.2.5 Retain

The retain stage represents the culmination of the CBR cycle, where newly solved cases are selectively incorporated into the case base. This careful curation process ensures the continuous evolution and enhancement of the system's problem-solving capabilities while maintaining the efficiency and relevance of the case base.

Our system employs a stringent evaluation process for case retention. A new case must satisfy at least one of the following criteria to be considered for inclusion. First, the case exhibits low similarity to existing cases in the case base, as determined by comparing feature vectors using our established similarity metrics. This criterion ensures that each retained case contributes unique information, preventing redundancy and maintaining the diversity of the case base. Second, the case has undergone human expert verification, providing a high degree of confidence in its correctness and relevance.

The retention strategy embraces both positive and negative experiences. Positive cases serve as valuable precedents for solving similar future problems, while negative cases function as critical learning examples, preventing the repetition of past errors. This balanced approach to retention aligns with the fundamental principles of case-based reasoning, as established by A [Aamodt and Plaza, 1994].

Each retained case incrementally enhances the system's problem-solving capabilities. As [Richter and Weber, 2013] notes, this iterative expansion of the case base exemplifies the essence of continual learning in CBR systems. With each cycle, the system becomes more adept at leveraging past experiences to address new challenges.

To maintain optimal performance, our system regularly evaluates the utility of retained cases, employs efficient indexing structures to manage the growing case base, and balances the trade-off between comprehensive coverage and computational efficiency.

6.3 Experiments and Results

We conducted our experiments using a computer equipped with a 3.60 GHz CPU, 16 GB of RAM, and a GTX 1060 GPU. The model implementation utilized TensorFlow [Abadi et al., 2016] and Keras frameworks.

6.3.1 Dataset

A significant challenge in phishing email detection research is the lack of a standardized, large-scale dataset for model development and comparison. Researchers typically rely on either open-source or proprietary datasets. Some of the most widely recognized datasets include the Enron Email Corpus [Enron email dataset, 2011], the SpamAssassin Public Corpus [Spam assassin project, 2015], the Nazario Phishing Corpus [Jose nazario phishing email corpus, 2004], and the Fraud Dataset [CLAIR collection of fraudemail, 2008].

To thoroughly assess the robustness and generalization capabilities of our proposed Deep Learning-based Case-Based Reasoning (DL-CBR) model, we employed two distinct datasets for evaluation.

Dataset 1 (D1) is a combination of three public corpora, which includes legitimate emails sourced from the Enron Corpus and the SpamAssassin Public Corpus (comprising both "easy ham" and "hard ham") as well as phishing emails collected from the SpamAssassin Public Corpus and the Nazario Phishing Corpus. The total composition of this dataset consists of 34,250 emails, with 28,137 being legitimate and 6,113 classified as phishing.

In contrast, Dataset 2 (D2) merges legitimate emails sourced solely from the Enron Corpus with fraudulent emails obtained from a dedicated fraud dataset. This dataset contains a total of 28,795 emails, including 25,596 legitimate emails and 3,199 phishing emails.

When comparing the two datasets, several key characteristics emerge. First, regarding source diversity, D1 combines data from three different corpora, whereas D2 is based on only two sources, which may lead to a greater degree of consistency in D2. Additionally, the class balance reveals a greater imbalance in D2, with a ratio of 8:1 for legitimate to phishing emails, compared to D1's ratio of 4.6:1. Finally, D2's reliance on a single source for legitimate emails may contribute to more consistent samples. Table 6.1 summarizes the composition of both datasets after removing duplicates.

Overall, by utilizing these two datasets, we aim to comprehensively evaluate our model's performance under diverse and realistic conditions, thereby enhancing its applicability in real-world scenarios.

Tableau 6.1: Description of Datasets Used in Phishing Email Detection Experiments [Remmide et al., 2024c]

Dataset	Legitimate	Phishing	Total
D1	28137	6113	34250
D2	25596	3199	28795

6.3.2 Experiment setup

In this experimental study, we employed a model utilizing pre-trained GloVe embeddings with 100 dimensions, followed by a contrastive representation learning network. The architecture includes a Bi-LSTM with 64 hidden units and a TCN with 32 hidden units, a kernel size of K=2, and dilation factors of d=[1,2,4,8,16]. This is complemented by an attention layer with a dropout rate of 0.2, and a dense layer containing 128 hidden units with tanh activation functions.

To facilitate objective evaluation, we implemented a series of baseline models, including AdaBoost, Decision Tree, LSTM, and CNN. Additionally, CBR approaches such as TF-IDF+CBR and Bag-of-words+CBR were compared with our DL-CBR model. TF-IDF and Bag-of-Words techniques were utilized to extract simple lexical features from the text, representing cases for the CBR case base. Each dataset was split into 90% training and 10% testing data using stratified randomization to maintain the original phishing-to-legitimate email ratio. To train the network, we implemented the N-pair multi-class loss function alongside the Adam optimizer, set with a learning rate of 0.001 for a duration of 100 epochs and L2 regularization of 0.001. The TCN applies ReLU activation. Additionally, we implemented a CBR classifier in Python. Table 6.2 summarises the parameters used in the experiment.

We conducted extensive experimentation with various configurations, testing different kernel sizes (k) of 2, 3, 4, and 5, and dilation factor sets, including d=[1,2,4], d=[1,2,4,8], and d=[1,2,4,8,16]. Adjustments were also made to the dimensions of GloVe, Bi-LSTM, and TCN. Ultimately, the hyperparameters presented yielded the most favorable outcomes, surpassing alternative configurations by a margin of 0.2% in terms of accuracy and precision.

Additionally, we implemented a comparative model using a different architecture: this model also utilized pre-trained GloVe embeddings with 100 dimensions, followed by a contrastive representation learning approach consisting of a CNN with 25 hidden layers, a max-pooling layer with a pool size of 2, and a Bi-LSTM with 50 hidden units. This architecture culminates in a dense layer of 100 units that represents the final output. To mitigate overfitting, we included a dropout of 0.3 between the third and fourth layers. This model was trained using contrastive loss and the Adam optimizer, followed by a CBR classifier implemented in Python.

To facilitate objective evaluation, we implemented a series of baseline models, including AdaBoost, Decision Tree, LSTM, and CNN. Additionally, CBR approaches such as TF-IDF+CBR and Bag-of-words+CBR were compared with our DL-CBR model. TF-IDF and Bag-of-Words techniques were utilized to extract simple lexical features from the text, representing cases for the CBR case base. Each dataset was split into 90% training and 10% testing data using stratified randomization to maintain the original phishing-to-legitimate email ratio.

Tableau 6.2: Parameters and Configurations Used in the Experimental Setup of phishing email detection

Parameter	**Value**
GloVe Embedding Dimensions	100
Bi-LSTM Hidden Units	64
TCN Hidden Units	32
TCN Kernel Size	2
TCN Dilation Factors	[1, 2, 4, 8, 16]
Dropout Rate	0.2
Dense Layer Hidden Units	128
Activation Function (Dense Layer)	tanh
Training Loss Function	N-pair multi-class loss
Optimizer	Adam
Learning Rate	0.001
Number of Epochs	100
L2 Regularization	0.001
TCN Activation Function	ReLU
Dropout Between Layers	0.3 (between 3rd and 4th layers)

6.3.3 Results and Discussion

Table 6.3 presents the performance of various machine learning (ML) and deep learning (DL) algorithms for phishing email detection on two datasets, D1 and D2.

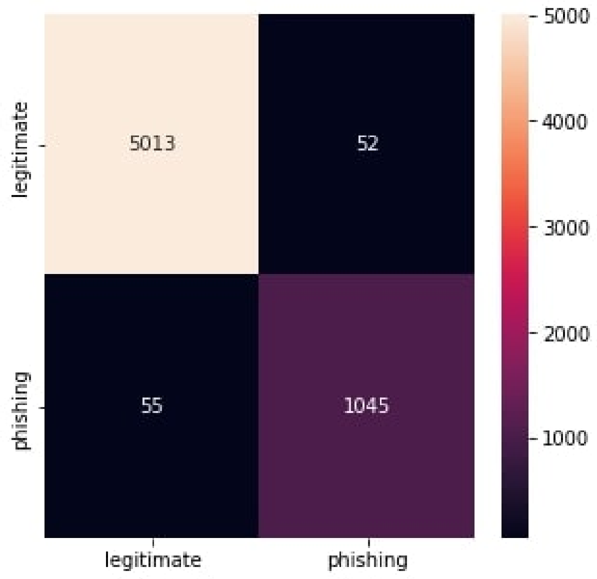
Tableau 6.3: Performance of ML and DL algorithms for the detection of phishing emails [Remmide et al., 2024c]

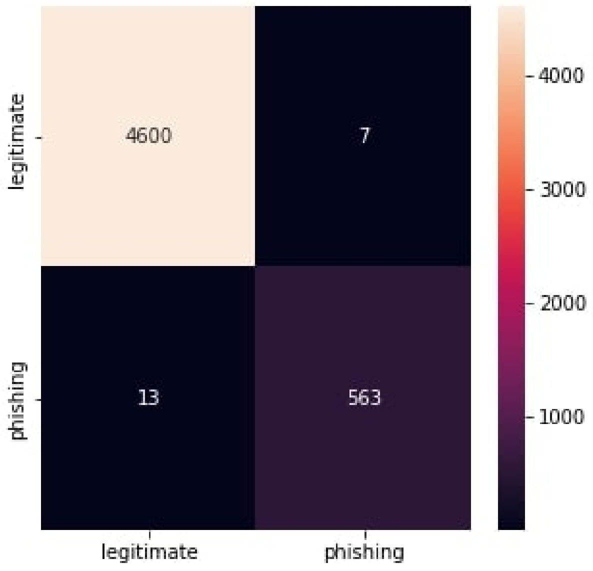
Dataset	model	Precision	Recall	Accuracy	F-measure
D1	AdaBoost	55.31	61	84.24	58.01
	Decision Tree	51.90	60.54	82.95	55.89
	Bi-LSTM	95.74	92	97.84	93.83
	CNN	92.86	94.72	97.76	93.78
	TF-IDF+CBR	61.50	39.36	84.78	48
	Bag-of-word+CBR	43.93	61.43	80.60	36.78
	CNN+bi-LSTM-CBR	95	95.56	96.18	95.01
	DL-CBR	**95.25**	**95**	**98.28**	**95.12**
D2	AdaBoost	57.37	66.14	90.77	61.45
	Decision Tree	56.36	66.14	90.54	60.86
	Bi-LSTM	99.81	92.70	99.17	96.12
	CNN	99.64	96.18	99.53	97.87
	TF-IDF+CBR	75.60	10.76	89.69	18.84
	Bag-of-word+CBR	68.75	3.81	89.11	7.23
	CNN+bi-LSTM-CBR	98.06	97.50	97.81	98.18
	DL-CBR	**98.71**	**97.74**	**99.61**	**98.25**

Our proposed DL-CBR model exhibited strong overall performance across both datasets, D1 and D2, consistently outperforming the best baseline model, Bi-LSTM. On D1, it achieved a low false positive rate of 0.96% and a high true negative rate of 99.03%, with precision at 95.25%, recall at 95%, accuracy at 98.28%, F-measure at 95.12%, and an AUC of 96.74%. For D2, the model's performance improved further, featuring a very low false positive rate of 0.13% and a high true negative rate of 99.86%, alongside impressive scores of precision at 98.71%, recall at 97.74%, accuracy at 99.61%, F-measure at 98.25%, and an AUC of 98.80%.

The true positive rate on D1 was 95%, while on D2 it was 97.74%, indicating the model's effectiveness in accurately identifying a significant majority of phishing emails. Additionally, the confusion matrix for our DL-CBR model, depicted in Figures 6.6a and 6.6b, shows high values for true positives (TP) and true negatives (TN), with minimal false positives (FP) and false negatives (FN). The model achieved an AUC of 98.79% and minimized the false negative rate to approximately 0.0225, underscoring its efficacy in phishing detection.

(a) Confusion Matrix Displaying Results from the Test Set of Dataset 1.





(b) Confusion Matrix Displaying Results from the Test Set of Dataset 2.

Figure 6.6: Confusion matrices [Remmide et al., 2024c]

These comprehensive results demonstrate the model's ability to reliably differentiate between legitimate and phishing emails. The high precision and recall reflect its proficiency in correctly classifying both categories, while the AUC scores indicate excellent discrimination, particularly in dataset D2, where classification approached perfection. The low false positive rates are encouraging, as misclassifying legitimate emails can pose significant operational challenges.

Although some phishing emails were missed, the model effectively avoided incorrectly flagging genuine emails as threats. The notably strong performance on dataset D2 emphasizes the model's generalization capabilities across diverse email distributions.

The challenges posed by dataset D1, with its varied legitimate email sources and multiple phishing techniques, contribute to a more robust evaluation of the model's performance. As shown in Table 6.3, our model outperformed other CBR models that utilized traditional text representation techniques, such as TF-IDF and Bag-of-Words.

In addition to comparing our model with the baseline models, we also compare it with published studies in terms of accuracy, precision, recall, and F-measure as shown in Table 6.4. Our DL-CBR model demonstrates competitive performance in phishing email detection. Notably, [Fang et al., 2019] achieved slightly better results; however, their method utilised both headers and body text for detection, while our model relies solely on the email body. Furthermore, while [Bountakas and Xenakis, 2023] achieved superior precision, recall, and F-measure, our model excels in accuracy, relying exclusively on textual features derived from semantic representations learnt from raw text through word embedding and sequence modelling layers.

Tableau 6.4: Comparison of the Proposed Model with Related Work in Phishing Email Detection [Remmide et al., 2024c]

Author	**Precision**	**Recall**	**Accuracy**	**F-measure**
[Fang et al., 2019]	99.66	99.00	99.84	99.33
[Alhogail and Alsabih, 2021]	98.5	98.3	98.2	98.55
[Halgaš et al., 2020]	97.45	95.98	96.74	96.71
[Halgaš et al., 2020]	-	98	96.74	97
[Nguyen et al., 2018]	97	95	99	96
[Bountakas and Xenakis, 2023]	99.43	99.43	99.43	99.42
DL-CBR	**98.71**	**97.74**	**99.61**	**98.25**

Furthermore, our DL-CBR model outperformed [Nguyen et al., 2018], [Halgaš et al., 2020], [HB et al., 2018], [Alhogail and Alsabih, 2021].

In addition to competing with published studies, our approach benefits from leveraging human expertise to rectify misclassified emails during the revision process. Correctly labeled instances from these misclassifications are reintegrated into the case base, enhancing the classifier's performance on similar future inputs by providing a richer set of examples for retraining. This iterative process, as highlighted in related work [Perner, 2019, LOPEZ DE MANTARAS et al., 2005, Avr, 2008], can improve system accuracy and adaptability over time, given the right conditions and high-quality data.

Incorporating feedback from human reviewers may also bolster our model's performance against zero-day attacks. The insights provided can aid in identifying and addressing emerging threats that have not been encountered previously, thereby enhancing the system's ability to detect and mitigate such risks.

While the manual revision method is valuable for specific scenarios with limited data, such as smaller, mission-critical environments, it may not scale effectively for larger datasets. Human review may hinder the system's ability to meet real-world demands. For larger-scale applications, a hybrid approach that combines manual revision with autonomous learning techniques may be necessary to ensure both accuracy and efficiency. This strategy could harness human expertise for critical tasks while enabling the system to learn and adapt independently as data volumes increase.

6.4 Conclusion

In this chapter, we introduced DL-CBR, a novel approach that combines Case-Based Reasoning with deep learning to effectively detect phishing emails. Utilizing pre-trained GloVe embeddings, a TCN, and a Bi-LSTM network, our model achieved impressive results, including a precision of 98.71% and low false negative rates across two datasets. Despite its strong performance, we recognized limitations in automation and case retention, suggesting future research directions, including semi-supervised learning techniques.

To address the critical issue of privacy in detecting phishing attempts, we will shift our focus to SMS phishing detection, or "smishing," utilizing federated learning. This approach enables us to enhance security while keeping sensitive user data on local devices, ensuring that personal information remains private.

Chapter 7

A Privacy-Preserving Approach for Detecting Smishing Attacks using Federated Deep Learning

7.1 Introduction

In the last few years, the expansion of mobile technology has allowed users unprecedented convenience for communication and information use. This increase in smartphone usage has simultaneously exposed people to new kinds of cyber threats, with smishing—using SMS messages for phishing attacks—emerging as a particularly threatening issue. In smishing, attackers exploit the trust that people have in SMS communication, tricking them into either revealing sensitive information or running malicious software. This phenomenon, first named by McAfee [Kang et al., 2014], is a company that excels in Internet security.

The analysis of potential threats in smishing attacks encounters unique challenges because of the restricted nature of mobile devices, the secretive nature of SMS content, and the imperative to maintain user privacy during the analysis. The traditional centralised technique for threat detection creates major privacy concerns since it typically collects user data in one location. As a direct result, there is an essential need for privacy-respective solutions that can monitor these attacks without infringing on user privacy.

This chapter presents a novel contribution to the field: an approach to privacy protection for the detection of smishing attacks leveraging Federated Deep Learning. By taking advantage of federated learning, this approach permits collaborative training of deep learning models across decentralised devices without transferring raw data to a central server. This maintains that sensitive data continues to stay securely on users' devices, therefore enhancing privacy and improving the recognition of smishing attacks.

7.2 Architecture

In developing an effective smishing detection system, we explored various learning models, including both unfederated and federated learning approaches.

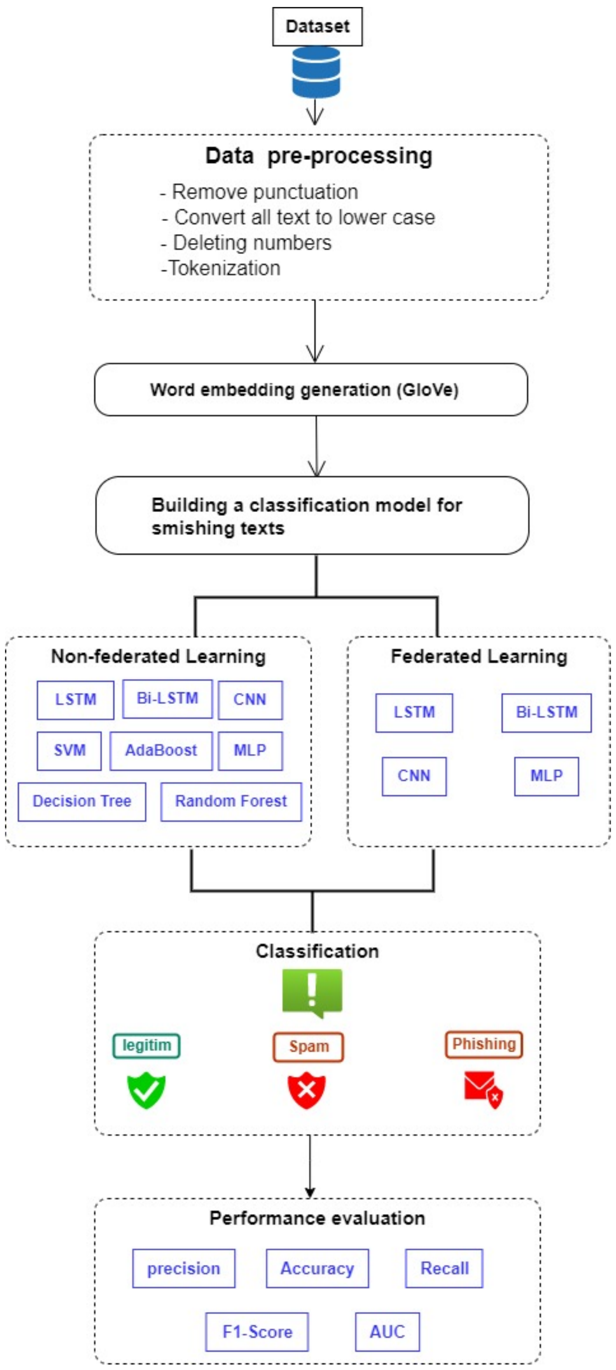


Figure 7.1: Pipeline of the smishing detection model

The overall architecture of the proposed solution is depicted in Figure 8.1. The approach starts with preprocessing the dataset, applying techniques like tokenisation, removing punctuation, and converting text to lowercase. During the next phase, we use the GloVe word embedding technique to turn the preprocessed text into a high-dimensional vector space. This vectorised representation is then provided to a classification model that classifies smishing messages apart from legitimate ones. The model is trained on the preprocessed data combined with GloVe embeddings. Finally, the performance of the classification model is evaluated using metrics like precision, recall, F1 score, and AUC to assess its effectiveness in detecting smishing messages.

Initially, we experimented with unfederated learning models, as shown in Figure 7.2. Different algorithms were involved: LSTM, CNN, SVM, Decision Tree, Random Forest, MLP, and AdaBoost. These models have been tested on separate data, yielding insights into their performance and limitations for smishing detection, based on metrics such as accuracy, precision, recall, and F1 score.

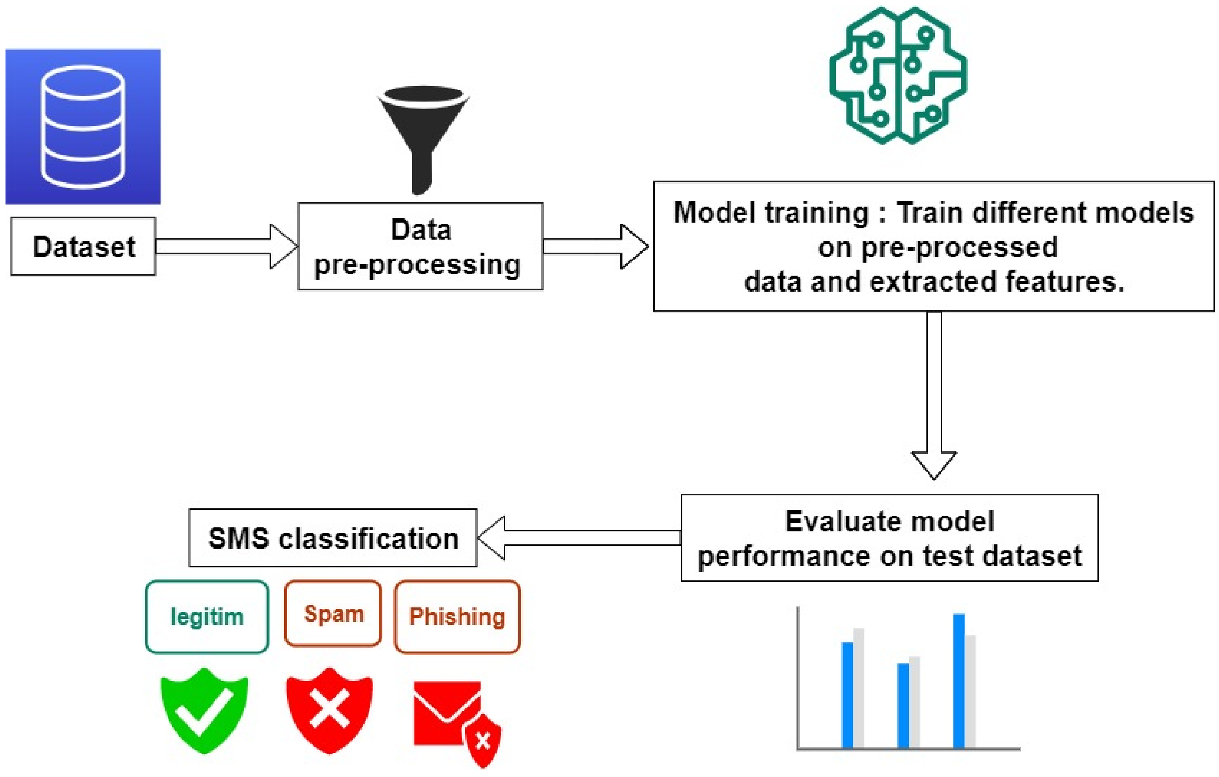

Dataset
Data pre-processing
Model training : Train different models on pre-processed data and extracted features.
Evaluate model performance on test dataset
SMS classification
legitim
Spam
Phishing

Figure 7.2: Architecture of the Unfederated Learning Model for Smishing Detection

After that, we shifted to a federated learning framework, shown in Figure 7.3. The process of federated learning enables distributed devices to collaboratively train a model without sharing raw data. In our setup, each device trains its own model locally, and only shares model parameters with a central server to uphold data privacy. This method successfully handles the issue of data distribution imbalance and privacy concerns while maintaining the model's performance.

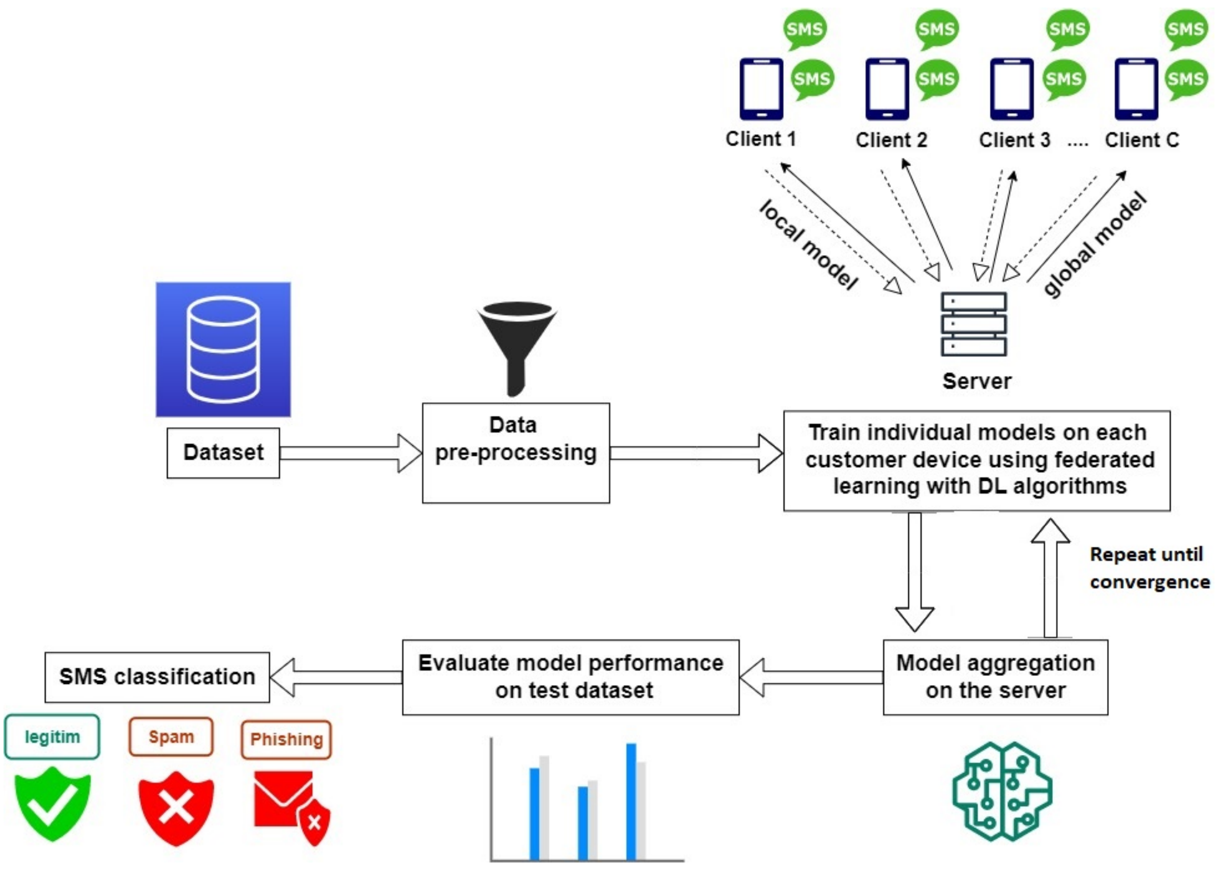

SMS
Client 1
Client 2
Client 3
....
Client C
local model
global model
Server
Dataset
Data pre-processing
Train individual models on each customer device using federated learning with DL algorithms
Repeat until convergence
Model aggregation on the server
Evaluate model performance on test dataset
SMS classification
legitim
Spam
Phishing

Figure 7.3: Federated learning model architecture for smishing detection [Remmide et al., 2024d]

7.3 Dataset

The SMS phishing dataset used in this study was developed by Sandhya Mishra and Devpriya Soni [Mishra and Soni, 2023]. It contains a total of 5,971 SMS messages, categorised into three classes: legitimate (Ham), spam, and phishing. More specifically, the dataset composes of 4,844 valid messages, 489 spam messages, along with 638 phishing messages. The illustration in Figure 7.4 reveals the distribution of SMS messages.

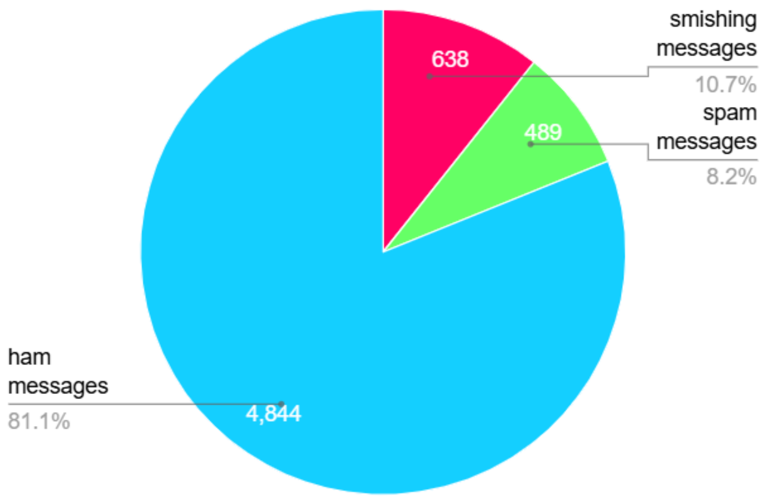


Figure 7.4: SMS phishing dataset distribution

7.4 Data Preprocessing and word embedding

The initial stages of our approach focus on preparing raw data for model training through preprocessing and feature extraction. These crucial steps ensure that the input data is appropriately formatted for machine learning models and that it captures the most relevant information for smishing detection.

Our preprocessing pipeline applies a series of techniques to clean and standardize the raw text data, improving both the quality and consistency of the input:

- **Tokenization**: The raw text is segmented into individual words or subword units, enabling the model to process linguistic information at a granular level.
- **Punctuation Removal**: All punctuation are removed to reduce noise, allowing the model to focus on meaningful textual content.
- **Lowercase Conversion**: The entire text corpus is converted to lowercase to ensure uniformity, preventing the model from treating identical words with varying cases (e.g., "Apple" vs. "apple") as distinct entities.

- **Stopword Removal**: Commonly used words that offer little semantic value (e.g., "the," "is," "and") are eliminated to reduce dimensionality and emphasize more informative terms.
- **URL and Phone Number Normalization**: URLs and phone numbers are replaced with standardized tokens, allowing the model to generalize these entities without being influenced by specific addresses or numerical patterns.

Following preprocessing, the cleaned text is transformed into numerical representations suitable for input into machine learning models. This transformation is performed using the Global Vectors for Word Representation (GloVe) algorithm, which generates high-dimensional vector representations for each word.

The output of this preprocessing and embedding pipeline is a set of fixed-length vector sequences. Each sequence represents an entire text message, where individual elements within the sequence correspond to high-dimensional vectors that capture both the semantic meaning of words and their contextual relationships within the message.

7.5 Building classification model

We study two methods for building classification model for smishing texts: unfederated learning and federated learning.

Unfederated learning

In the unfederated learning approach, we experimented with several well-established machine learning and deep learning models, including LSTM, Bi-LSTM, CNN, SVM, Decision Tree, Random Forest, MLP, and AdaBoost. Various performance metrics were used to evaluate the effectiveness of each model.

Federated Learning

The second approach utilizes federated learning, which allows the training of models on decentralized data sources without sharing the raw data between devices. In this framework, we focus on deep learning models, specifically LSTM, Bi-LSTM, CNN, and MLP.

We chose these algorithms due to their ability to handle the complex patterns and relationships found in textual smishing data. Deep learning algorithms excel at identifying intricate patterns and connections in data using multiple layers of neural networks. This is particularly advantageous for smishing detection, as text messages often contain subtle linguistic and contextual information that deep learning models can better capture.

Algorithm 3 presents an overview of the federated learning process for smishing detection, which consists of a coordinating server and several client devices. Each client, represented by C, where C ∈ [1, 2, 3,4], has its own set of local SMS data.

Algorithm 3 Federated Learning for Smishing Detection [Remmide et al., 2024d]

```
Input: Smishing dataset
Output: Model performance (e.g., Accuracy, F1-score, and Precision)
/* Server-side */
Server: Initialize and send global model $W_t$ to all $C$ clients
for each epoch $e \in E$ do
    for each client $c \in \{1, 2, ..., C\}$ in parallel do
        $W_{c_t} \leftarrow \text{ClientUpdate}(W_{c_t})$  //local updates
    Perform weighted averaging and update the global model: $W_{t+1} \leftarrow \sum \left(\frac{n_c}{n}\right) \cdot W_{c_t}$, where
$n_c$ is the number of samples in client $c$ and $n$ is the total number of samples
    Send the updated global model $W_{t+1}$ to all clients
/* Client-side at each client $c$ */ ClientUpdate($W_{c_t}$):
/* Runs once at the beginning */
Prepare smishing dataset:
  • Data extraction
  • Data preprocessing
  • Perform an 80:20 train-test split
  • Tokenization
/* Runs repetitively during training/testing */ whileglobal model $W_t$ is received
from the server do
    Set $W_{c_t} = W_t$
    /* Training/testing on the local smishing dataset $X$ with $n_c$ samples */
    for each batch $b \in B$ do
        $W_{c_t} \leftarrow W_{c_t} - \eta \cdot O_f(W_{c_t}; X)$, for $X \sim P_c$
        /* $B$ is the batch size, $\eta$ is the learning rate, and $O_f(W_{c_t}; X)$ represents
gradients with respect to the cost function */
    Send locally trained $W_{c_t}$ to the server for aggregation
```

The federated learning process proceeds as follows:

1. **Local Training:** Each client device trains a local model using its smishing dataset D_C. After training, the model on each client c is updated to WC_t. This training is performed independently on each client's local data without sharing the raw data.

2. **Model Aggregation:** After each client updates its local model, the parameters are sent to the central server. The server then performs aggregation by calculating a weighted average. This process results in an updated global model, denoted as W_{t+1}. The aggregation step can be mathematically represented by equation as shown in Eq. 7.1:

$$\theta_{\text{global}} = \frac{1}{N}\sum_{i=1}^{N}\theta_i \tag{7.1}$$

   Here, θ_{global} denotes the global model parameters, N represents the number of participating clients, and θ_i corresponds to the model parameters of the i-th client. The averaging ensures that each client's contribution is proportionally incorporated into the global model.

3. **Global Model Distribution:** After aggregation, the updated global model W_{t+1} is distributed back to all participating clients, ensuring that all clients work with the most recent version of the model. This synchronization facilitates collaborative learning among clients.

4. **Iterative Training:** The process of local training, model aggregation, and synchronization is repeated iteratively over multiple epochs until the global model converges and achieves optimal performance for smishing detection.

This federated learning setup ensures data privacy by keeping the raw SMS datasets on the client devices. Instead of transmitting sensitive user data, only model parameters are exchanged between clients and the central server, ensuring that personal information remains secure and confidential throughout the training process. This approach is particularly suited for scenarios where user privacy is paramount, such as mobile service providers collaborating on smishing detection without exposing individual users' SMS content.

Table 7.1 details the optimized hyperparameters of the proposed approach.

Tableau 7.1: Hyperparameters for Models in Smishing Detection

Model	**Hyperparameter**	**Value**
LSTM, Bi-LSTM CNN, MLP	Batch Size	64
	Dropout Rate	0.3 (LSTM), 0.4 (Bi-LSTM, MLP)
	Optimizer	Adam
	Epochs	50 (LSTM, Bi-LSTM), 30 (CNN, MLP)
LSTM	LSTM Units	100
	Layers	2
Bi-LSTM	LSTM Units (per direction)	128
	Layers	2
CNN	Conv. Layers	3
	Filter Sizes	[64, 128, 256]
	Kernel Size	3x3
MLP	Hidden Layers	2
	Neurons (per layer)	[128, 64]
SVM	Kernel	Radial Basis Function (RBF)
	Regularization (C)	1.0
	Gamma	Scale
	Max Iterations	1000
Decision Tree	Max Depth	20
	Min Samples Split	2
	Min Samples Leaf	1
	Criterion	Gini
Random Forest	Number of Trees (n_estimators)	100
	Max Depth	20
	Min Samples Split	2
	Criterion	Gini
AdaBoost	Number of Trees (n_estimators)	50
	Base Classifier	Decision Tree
	Max Depth of Base Estimator	1

7.6 Results

In this study, we evaluated the performance of several deep learning and traditional machine learning models for smishing detection. Both centralized (unfederated) and decentralized (federated) learning approaches were tested. Performance was assessed using standard metrics: accuracy, precision, recall, and F1-score, providing a comprehensive assessment of each model's performance.

7.6.1 Unfederated learning

In our study, we evaluated different deep learning and machine learning models to detect smishing in our dataset, such as: LSTM, Bi-LSTM, CNN, SVM, Decision Tree, Random Forest, MLP and AdaBoost.

Deep learning algorithms :

Table 7.2 summarises the performance of the implementation of the four deep learning models: LSTM, Bi-LSTM, CNN, and MLP in the centralised approach.

Tableau 7.2: Performance of Deep Learning Models (Unfederated Learning) [Remmide et al., 2024d]

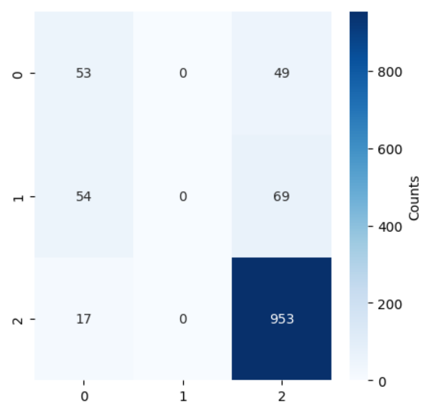
Algorithm	Accuracy	Precision	Recall	F1-Score
LSTM	87.86%	81%	88%	84%
Bi-LSTM	**92.3%**	**92%**	**91%**	**91%**
CNN	91.54%	91%	92%	91%
MLP	80.5%	69%	81%	74%

The results show that the Bi-LSTM model achieved the highest accuracy at 92.30%. The CNN model followed closely behind with an accuracy of 91.54%. Both LSTM and MLP models showed lower performance compared to Bi-LSTM and CNN, with accuracies of 87.86% and 80.5%, respectively.

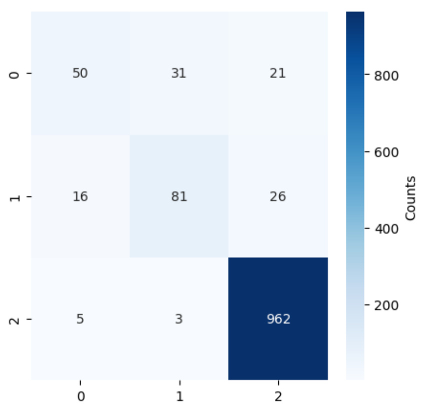
The confusion matrices of these deep learning models, as illustrated in Figure 7.5, provide further insights into their classification capabilities.

Machine learning algorithms :

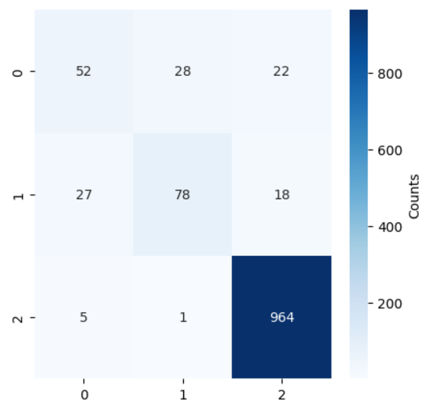
In addition to deep learning models, we evaluated four traditional machine learning algorithms: SVM, Decision Tree, Random Forest, and AdaBoost. The results of these evaluations are presented in the following table 7.3:



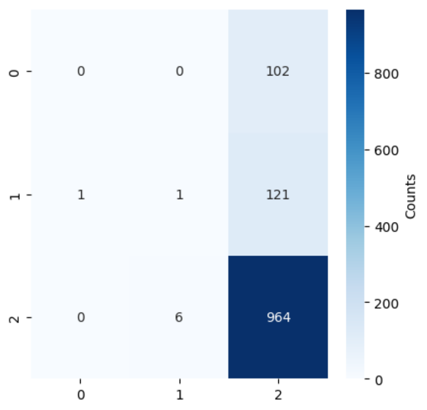
(a) Confusion Matrix Representing the Performance of the LSTM model



(b) Confusion Matrix Representing the Performance of Bi-LSTM model



(c) Confusion Matrix Representing the Performance of the CNN model



(d) Confusion Matrix Representing the Performance of MLP model

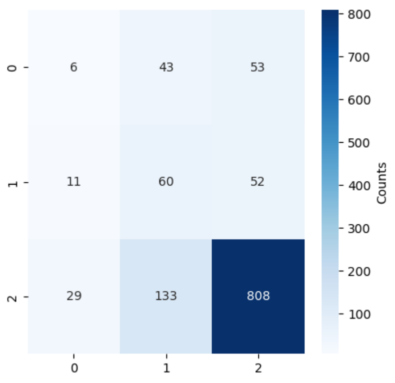
Figure 7.5: Confusion Matrices Showing Performance for LSTM, Bi-LSTM, CNN and MLP modelss

Tableau 7.3: Performance of Machine Learning Models (Unfederated Learning)

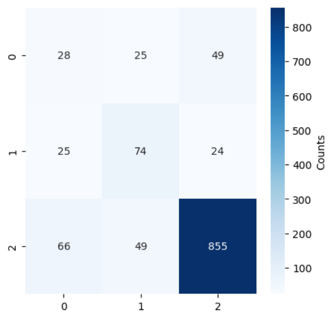
Algorithme	Accuracy	Precision	Recall	F1-Score
SVM	73.13%	76%	73%	74%
Decision Tree	80.08%	82%	80%	80 %
Random Forest	**88.95%**	**87%**	**89%**	**87%**
AdaBoost	86.61%	85%	87%	86%

Among the machine learning algorithms, Random Forest demonstrated the best performance with an accuracy of 88.95%, followed by AdaBoost with an accuracy of 86.61%. Decision Tree performed moderately well with an accuracy of 80.08%, while SVM showed the lowest performance with 73.13% accuracy.

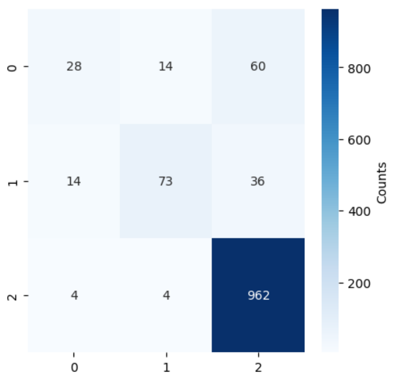
The confusion matrices for each algorithm are presented in Figure 7.6:



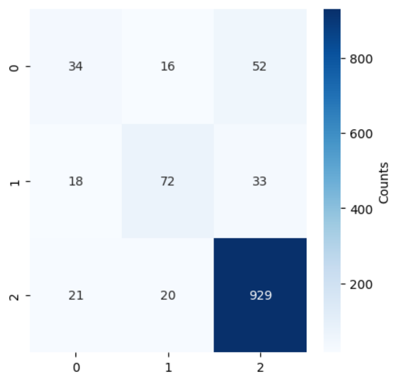
(a) Confusion Matrix Representing the Performance of the SVM Model



(b) Confusion Matrix Representing the Performance of the Decision Tree Model



(c) Confusion Matrix Representing the Performance of the Random Forest Model



(d) Confusion Matrix Representing the Performance of the AdaBoost Model

Figure 7.6: Confusion Matrices Showing Performance for SVM, Decision Tree, Random Forest, and AdaBoost Models

The SVM model Figure 7.6a exhibited the lowest classification accuracy, as seen from its relatively high rate of false positives and false negatives. The Decision Tree model Figure 7.6b performed moderately well, though misclassifications were still present. In contrast, the Random Forest model Figure 7.6c achieved the best performance among the machine learning algorithms, with fewer misclassifications, especially in distinguishing smishing messages. The AdaBoost model Figure 7.6d showed similar classification strength, producing balanced metrics and minimizing misclassifications.

7.6.2 Federated Learning

In the federated learning context, we applied the same deep learning models (LSTM, Bi-LSTM, CNN, MLP) to evaluate their performance. The results of our federated learning experiments are presented in the following table 7.4 :

Tableau 7.4: Measures for Assessing the Performance of Federated Learning Models

Algorithm	Accuracy	Precision	Recall	F1-Score
LSTM	81.17%	66%	81%	73%
Bi-LSTM	88.78%	88%	89%	87%
CNN	**92.38%**	**92%**	**92%**	**92%**
MLP	88.87%	88 %	89%	87%

In the federated learning setup, the CNN model achieved the highest accuracy, with 92.38%, surpassing all other models. Both the Bi-LSTM and MLP models performed well, with accuracies of 88.78% and 88.87%, respectively. The LSTM model, however, exhibited the lowest performance, with an accuracy of 81.17%.

The confusion matrices for the federated learning models are shown in Figure 7.7. The CNN model Figure 7.7c continued to demonstrate superior performance, with very few misclassifications, showcasing its ability to generalize well across the dataset. The Bi-LSTM model Figure 7.7b also performed strongly, though it had slightly more misclassifications than CNN. Both the MLP and LSTM models Figures 7.7d and 7.7a struggled more in the federated context, as reflected by their lower classification accuracy.

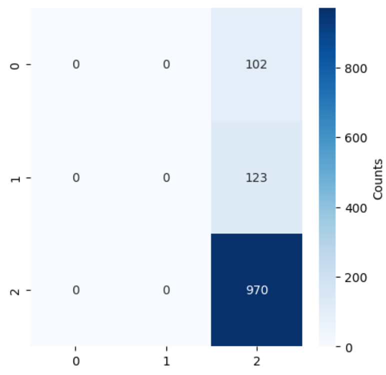
7.6.3 Synthesis

Our study provides valuable insights into the effectiveness of both federated and non-federated learning approaches for smishing detection. Comparing the two approaches reveals that the best-performing model in federated learning, CNN with an accuracy of 92.38%, slightly outperformed the best model in non-federated learning, Bi-LSTM with an accuracy of 92.3%. This demonstrates the potential of federated learning for smishing detection while preserving data privacy.

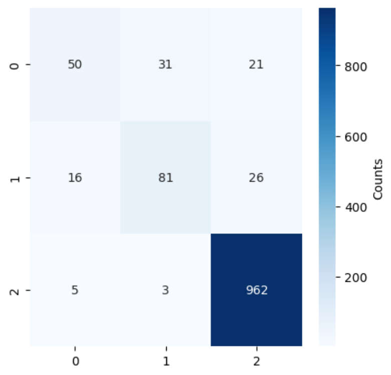
To contextualize our results, we compared them with previous studies using the same dataset, as illustrated in Table 7.5.

Tableau 7.5: Comparison of Results Between the Proposed Method and Various Classification Techniques [Remmide et al., 2024d]

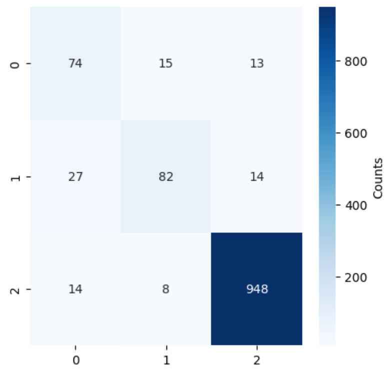
Research	Algorithm	Precision	Accuracy	Recall	F1-Score	AUC
[Mishra and Soni, 2021]	Backpropagation Algorithm	84%	97.93%	94%	/	98.8%
[Mishra and Soni, 2022]	Naive Bayes	93%	91.6%	92%	92%	/
[Mishra and Soni, 2022]	ANN	/	94%	/	86.72%	/
Our approach	**CNN**	**92%**	**92.38%**	**92%**	**92%**	/



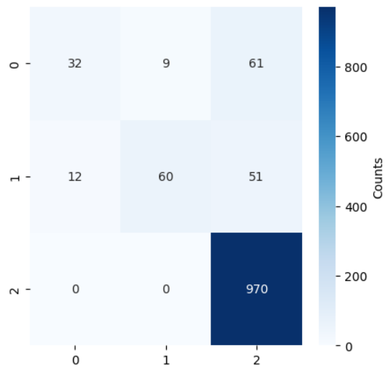
(a) Confusion Matrix Representing the Performance of the LSTM model



(b) Confusion Matrix Representing the Performance of Bi-LSTM model



(c) Confusion Matrix Representing the Performance of the CNN model



(d) Confusion Matrix Representing the Performance of MLP model

Figure 7.7: Confusion Matrices Showing Performance for federated LSTM, Bi-LSTM, CNN and MLP models

Our federated learning approach with CNN (92.38% accuracy) performed competitively compared to centralized approaches in the literature. While the accuracy is slightly lower than that reported in some previous studies, our method offers significant advantages in terms of data privacy and security. As each client device in our federated learning setup trained its own local model with its own private data, and only model updates were shared with the central server for aggregation. This decentralized approach ensures that user data remains secure and private, mitigating the privacy issues associated with central data storage. This makes it particularly well-suited for real-world applications where data privacy is of paramount importance.

7.7 Conclusion

This study has made a substantial contribution to the field of smishing detection by leveraging federated learning to enhance data privacy. Through a comprehensive analysis of various machine learning and deep learning models, we demonstrated the efficacy of federated learning in addressing the challenges of privacy-preserving smishing detection.

Notably, our CNN model, combined with federated learning, achieved an average accuracy of 92.38%, marking a significant outcome of the study. While this accuracy did not surpass all existing benchmarks, our primary focus was on balancing performance with privacy, ensuring that user data remained decentralized and secure. This represents a significant step forward in the development of privacy-conscious cybersecurity solutions.

In the next chapter, we extend our investigation into phishing detection by employing authorship verification techniques. This approach focuses on identifying phishing attacks based on writing style analysis, further enhancing our ability to detect social engineering threats while safeguarding user privacy.

Chapter 8

Authorship Verification for Phishing Email Detection

8.1 Introduction

In the contemporary digital landscape, the proliferation of online communication channels has fundamentally transformed interpersonal and organizational relationships, facilitating rapid information exchange while simultaneously creating fertile ground for cybercrime [Parvinder, 2017]. Among various cyber threats, phishing attacks represent a particularly insidious form of social engineering, wherein attackers fraudulently obtain confidential information by manipulating victims into revealing financial details, passwords, or personal information [Awan, 2020].

The efficacy of phishing attacks stems from their exploitation of human psychological vulnerabilities rather than technical system weaknesses. A seminal study by [Benenson et al., 2017] demonstrated that approximately 20% of recipients clicked on specious links embedded in phishing emails, highlighting the human factor in cybersecurity breaches. This vulnerability has led to an unprecedented surge in phishing incidents, with the APWG reporting nearly five million attacks in 2023 marking it as the most severe year for phishing activities on record [APWG, 2023].

This chapter explores the theoretical foundations, methodological approaches, and empirical evaluation of authorship verification techniques in the context of phishing detection.

8.2 Architecture

This chapter presents a robust method for authorship verification through a Siamese deep learning network, designed to effectively classify emails into four distinct categories: phishing, legitimate, harassment, and suspicious.

In this methodology, email classification is redefined as an authorship verification task, based on the premise that each category possesses distinct stylistic and linguistic characteristics, regardless of any impersonation or attempts to obscure identity. Unlike traditional classification techniques that depend on specific indicators, this method utilizes the inherent writing style to ensure accurate email classification.

The classification process commences upon receiving a new email. A Siamese network, featuring three shared LSTM layers, encodes the incoming email into a vector representation. The network then calculates a similarity score between this vector and the representative vectors from each category, employing the Manhattan distance metric. These representative emails serve as reference points for evaluating similarity.

To identify these representative emails, the k-means clustering algorithm groups similar emails, forming distinct clusters for each category. This clustering assists in classifying incoming emails by comparing them to these representative examples. If the similarity score surpasses a predetermined threshold, the email is confidently categorized accordingly. Figure 8.1 illustrates the overall architecture of this solution.

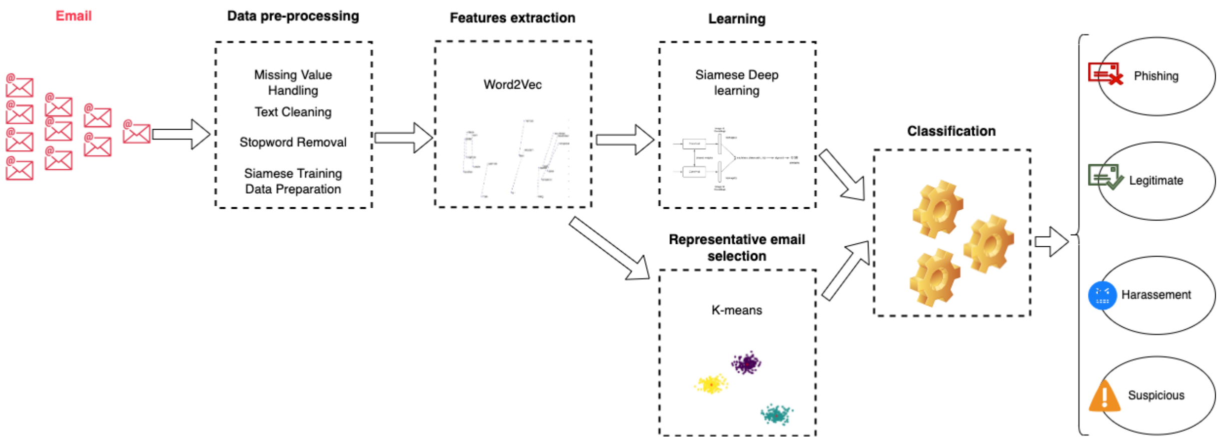


Figure 8.1: Workflow for the classification Phishing Emails [Remmide et al., 2024a]

8.2.1 Data pre-processing

The preprocessing phase addresses various challenges arising from the noisy and unstructured characteristics of email data. Our dataset often includes elements like HTML tags, URLs, mentions, emojis, and contractions, which can negatively impact the effectiveness of our classification model.

To mitigate these issues, we implement a thorough preprocessing pipeline that includes the following essential steps:

- **Handling Missing Values:** We remove any rows with missing values to ensure that the model receives complete and consistent data inputs.
- **Text Cleaning:** A series of text preprocessing functions are applied to minimize noise and transform the text into a more manageable format for classification. This process includes converting text to lowercase, eliminating Unicode characters, substituting URLs and mentions with placeholders, removing non-alphabetic characters and emojis, and expanding contractions (e.g., converting "can't" to "cannot").
- **Stopword Removal:** After text cleaning, common English stopwords are removed. Since these words generally do not contribute to the classification task, their elimination improves processing efficiency.
- **Preparation for Siamese Network Training:** The cleaned email text is then classified into categories to create sample pairs for training the Siamese network. This involves selecting subsets of samples for each class and generating positive pairs (samples from the same class) and negative pairs (samples from different classes) to enhance the network's ability to differentiate between categories.

8.2.2 Representative Email Selection

A critical component of our methodology is the identification of representative emails for each category (e.g., phishing, legitimate, harassment, suspicious). To achieve this, we employ k-means clustering, an unsupervised learning algorithm, to partition the training dataset into distinct clusters within each category. The cluster centroids are then selected as representative exemplars.

This approach offers several key advantages. First, it ensures comprehensive coverage of stylistic variations within each category by capturing the diverse linguistic and stylistic features present in the data. This allows the selected representatives to reflect the full range of characteristics within each category, improving the model's ability to generalize. Second, it enhances computational efficiency during the similarity assessment phase. By comparing emails to centroids rather than all other emails, the number of comparisons is significantly reduced, making the process faster and more scalable. Third, it provides robustness against outliers or anomalous samples, as the centroids are calculated based on the average of all points within a cluster, ensuring that the representatives are not skewed by rare or irregular data points.

The clustering process is mathematically defined as:

$$C \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 \tag{8.1}$$

where C represents the set of clusters, k is the number of clusters, x represents an email in cluster C_i, and μ_i is the centroid of cluster C_i.

The goal of k-means clustering is to minimize the within-cluster sum of squares, which measures the total squared distance between each email and its corresponding cluster centroid. By minimizing this value, the algorithm ensures that emails within the same cluster are as similar as possible, while emails in different clusters are as distinct as possible. The process begins with the random selection of k initial centroids, followed by iterative steps of assigning emails to the nearest centroid and recalculating the centroids until convergence is achieved.

Once the clusters are formed, the centroids (μ_i) are selected as the representative emails for each category. These centroids serve as exemplars that encapsulate the key characteristics of their respective clusters, enabling efficient and accurate similarity comparisons during the classification process. By selecting centroids from multiple clusters, we ensure that the representative emails cover a wide range of stylistic and linguistic variations within each category. This diversity is crucial for capturing the nuances of different email types. Additionally, the use of centroids improves computational efficiency and scalability, as the number of comparisons is significantly reduced. The centroids also provide robustness against outliers, as they represent the "average" characteristics of their clusters, making them reliable representatives that are less affected by noisy data.

8.2.3 Siamese model

The proposed method utilizes a Siamese network architecture, which is particularly effective for tasks that require similarity assessment. The architecture of the network is depicted in Figure 8.2.

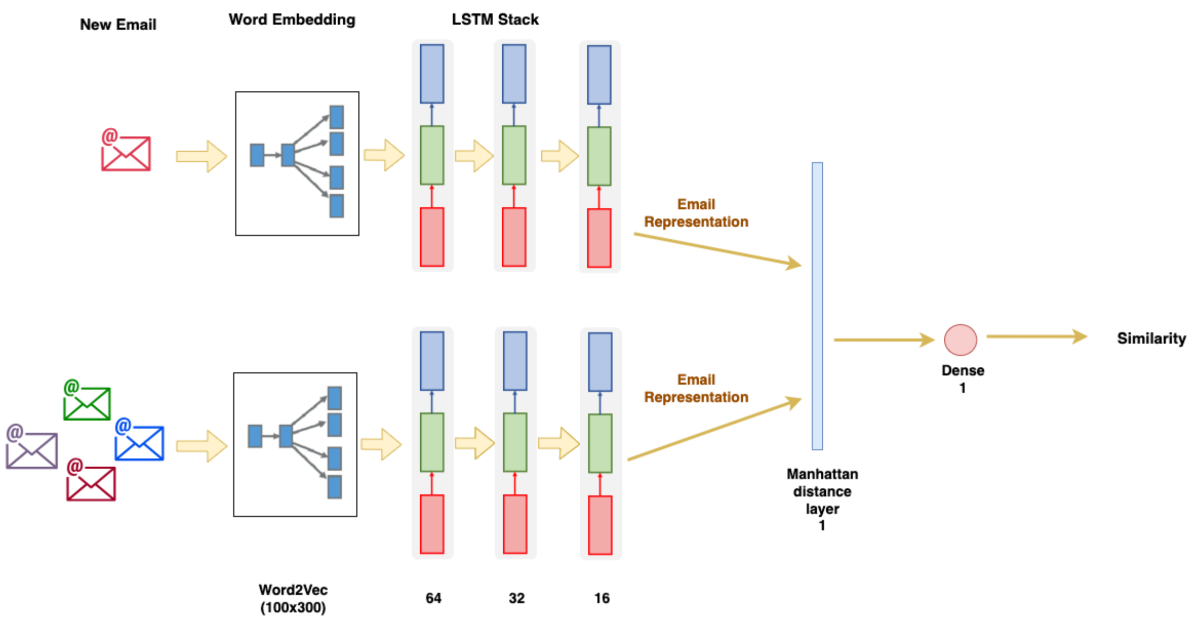


Figure 8.2: Siamese LSTM-Based Similarity Model for Email Classification [Remmide et al., 2024a]

This Siamese network consists of two input branches, each designed to accept an email sample. Initially, these email samples are transformed into dense vector representations through Word2Vec, resulting in vectors of 300 dimensions.

These vectors are then processed through a shared LSTM network, which encodes the email content into meaningful representations. The shared LSTM features three layers with progressively decreasing units—64, 32, and 16—allowing the model to capture the linguistic and stylistic intricacies present in the email data.

After passing through the shared LSTM network, each branch generates an output corresponding to its respective email sample. The similarity between the two output representations is computed using the negative Manhattan distance exponent, as indicated in Eq. 8.2:

$$Similarity = \exp(-\sum |x_i - y_i|) \tag{8.2}$$

The Manhattan distance (also known as the L1 distance) was chosen for several reasons. First, it is robust to outliers, as it sums the absolute differences between corresponding elements, making it less sensitive to extreme values compared to the Euclidean distance (L2). Second, it provides a straightforward and interpretable measure of similarity, which is particularly useful for tasks like email classification where understanding the model's decision-making process is important. Third, it is computationally efficient, making it suitable for large-scale datasets and real-time applications. The use of the exponential function further enhances the differences in similarity scores, helping the model better distinguish between emails from the same class (high similarity) and those from different classes (low similarity).

The training of the Siamese network employs the mean squared error (MSE) loss function, along with the Adadelta optimizer and gradient clipping. During training, the model learns to effectively encode the content of emails and distinguish between samples belonging to the same class (high similarity) and those from different classes (low similarity). This enables the Siamese network to identify patterns within the email data, facilitating accurate classification into four categories: phishing, legitimate, harassment, and suspicious.

8.2.4 Classification

The process of classifying a new email using the trained Siamese network involves the following steps:

1. **Determining Representative Vectors:** To establish representative vector representations for each email category, the output vector representations from the shared LSTM network are clustered using the K-means algorithm. The centroids of these clusters are designated as the representative vectors, as they encapsulate the essential stylistic and linguistic features that define each email category. These centroids, specific to each class, are stored as reference points for future email classification during the inference stage.

2. **Calculating Similarity:** The new email sample is processed through one branch of the Siamese network, while the other branch concurrently processes the representative vectors corresponding to each of the four classes: phishing, legitimate, harassment, and suspicious. The network computes similarity scores between the new email and each of these representative vectors.

3. **Assigning a Class Label:** The new email is assigned a class label based on which similarity score is the highest. For instance, if the highest score corresponds to the "phishing" category, the email is classified as phishing. This process is similarly applied to all classes, with the email being categorized into the class that has the highest similarity score.

4. **Applying a Confidence Threshold:** To improve classification accuracy, a confidence threshold is introduced. If the highest similarity score falls below this threshold, the system flags the new email as "uncertain" or "unclassified" instead of assign-

ing it to a specific category. This strategy helps reduce misclassifications, especially for emails that do not closely align with any of the predefined classes.

8.3 Results

This section provides details on the dataset used, the experimental setup, and the results obtained. The experiments were run on a system with a 2.11 GHz CPU, 32 GB RAM, and an RTX 3070 Ti GPU. The model was built using TensorFlow and Keras frameworks.

8.3.1 Dataset

Our experimental evaluation employed two distinct datasets to ensure robust validation of the proposed methodology:

- **Twitter Corpus**: A curated dataset from Kaggle comprising 26,000 tweets from 13 randomly selected celebrities, with each author contributing 2,000 tweets. This dataset provided a controlled environment for initial model evaluation, offering diverse writing styles while maintaining consistent authorship.

- **SeFACED Dataset** [Hina et al., 2021]: A comprehensive email corpus containing 32,427 messages across four categories: normal, fraudulent, harassment, and suspicious. This dataset integrates content from multiple sources, including the Enron Corpus, Phished Emails Corpus, and Hate Speech and Offensive Language Collection. Figure 8.3 illustrates the class distribution within this dataset. Fig8.3 presents the distribution of the different classes.

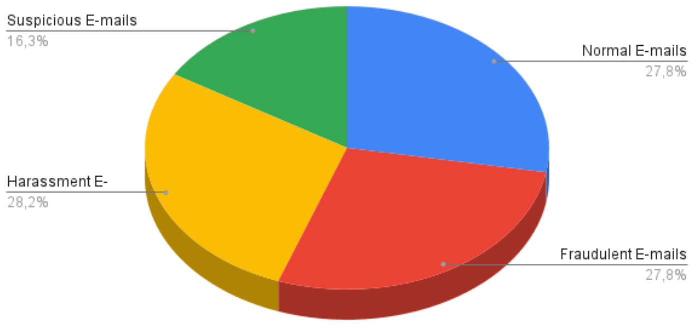


Figure 8.3: Overview of Email Class Distribution in the Dataset [Remmide et al., 2024a]

The utilization of two distinct datasets enabled us to evaluate our model's generalizability across different domains and writing styles.

8.3.2 Experimental Results and Analysis

Figure 8.4 presents a comparative analysis of our model's performance across both datasets, demonstrating robust results, with slightly superior metrics on the Twitter dataset. This performance difference can be attributed to factors such as dataset characteristics (e.g., consistency in writing style and text length), class distribution and balance, and domain-specific linguistic features. The model achieved accuracy rates of 97.12% and 90.12% on the Twitter and SeFACED datasets, respectively, indicating strong generalization capabilities across different textual domains.

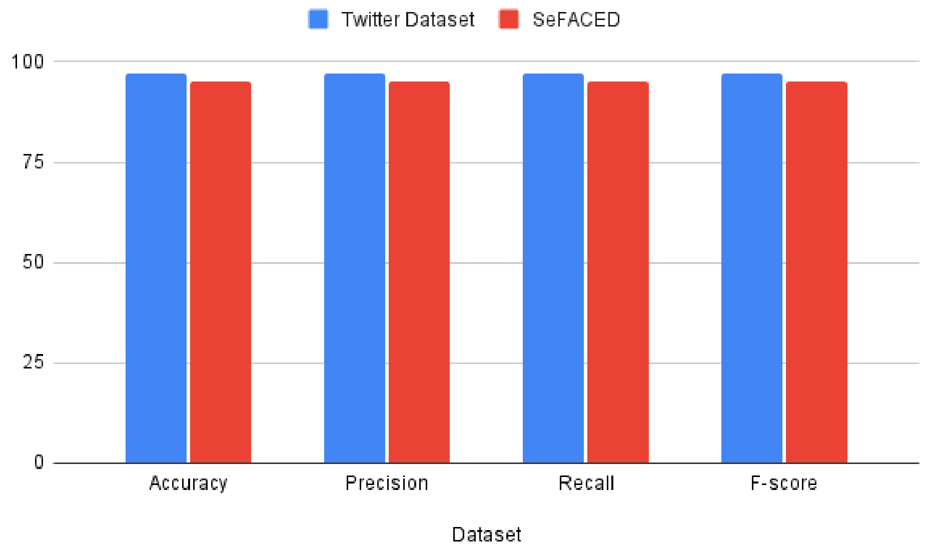


Figure 8.4: Comparative Performance of Authorship Verification Models on Twitter and SeFACED Datasets [Remmide et al., 2024a]

Our methodology's effectiveness was evaluated against leading approaches in both authorship verification and email classification:

Figure 8.5 illustrates that the models proposed in this approach attained an accuracy of 97.12%, which is on par with the HRSN model by [Boenninghoff et al., 2019] at 97% accuracy. It significantly outperformed the PRNN model [Hosseinia and Mukherjee, 2018], which reached 90% accuracy. Additionally, our method demonstrated competitive results against various architectural approaches, such as convolutional models [Shrestha et al., 2017] and feature-based systems [Weerasinghe and Greenstadt, 2020].

Figure 8.5 demonstrates that the models developed in this study produced results comparable to those of other leading research in authorship verification [Boenninghoff et al., 2019, Hosseinia and Mukherjee, 2018, Shrestha et al., 2017, Weerasinghe and Greenstadt, 2020]. Our method attained an accuracy of 97.12%, which closely aligns with the findings of [Boenninghoff et al., 2019], which reported an accuracy of 97% using an HRSN model.

Additionally, our results surpass those of [Hosseinia and Mukherjee, 2018], which achieved 90% accuracy with a PRNN model.

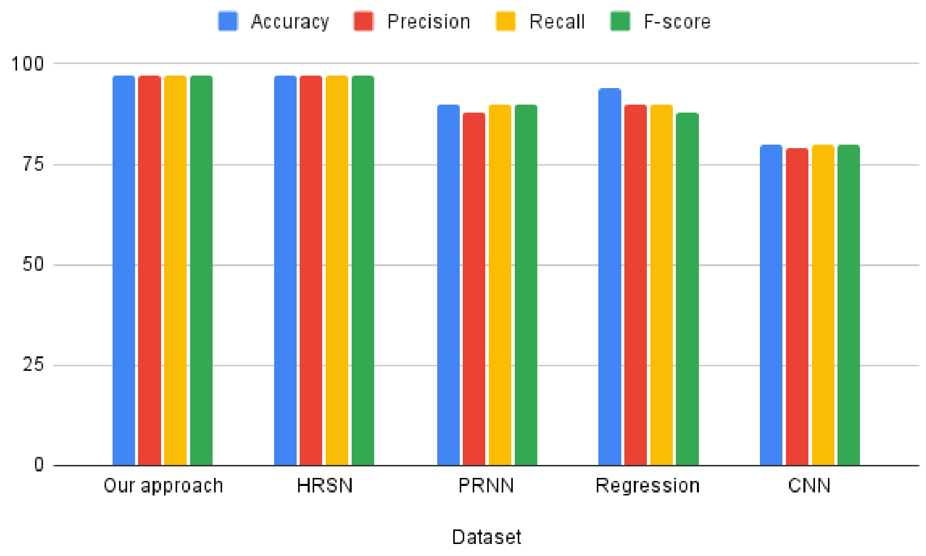


Figure 8.5: Performance assessment of the proposed models compared to leading research in authorship verification. [Remmide et al., 2024a]

In the email classification task (Figure 8.6), our model achieved an accuracy of 90.12% on the SeFACED dataset, showcasing strong performance. While this result is slightly below the original SeFACED approach [Hina et al., 2021], which achieved 95% accuracy, our model remains competitive within this domain.

The slight performance gap in email classification can be attributed to several factors: First, email content is inherently more complex than social media text, presenting unique challenges. Secondly, distinguishing between closely related email categories can be challenging due to overlapping characteristics. Finally, variations in email length may impact feature extraction, further complicating classification.

Overall, our methodology demonstrates robust performance across different domains, illustrating its versatility and effectiveness in both authorship verification and email classification tasks.

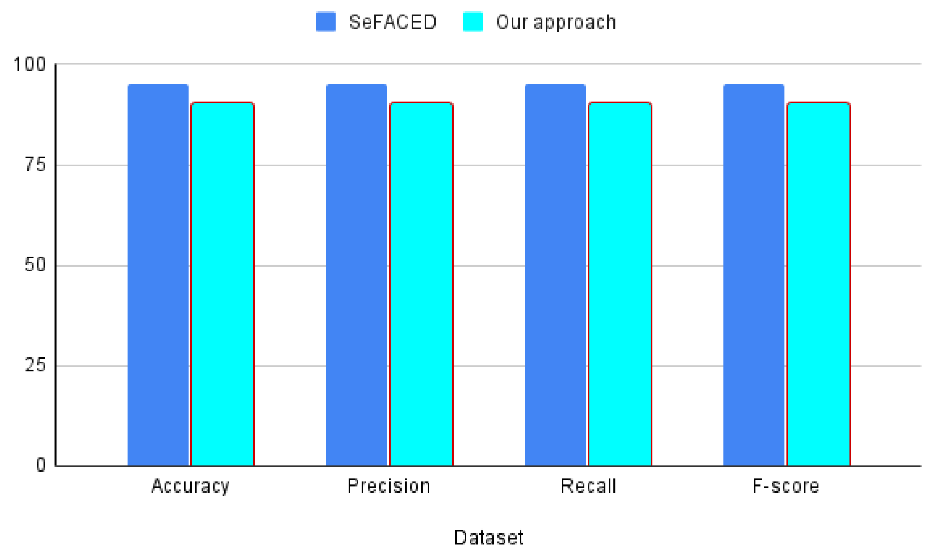


Figure 8.6: Performance comparison of the proposed models against leading research in email classification. [Remmide et al., 2024a]

8.4 Conclusion

In this chapter, we presented an innovative method for email classification that employs a Siamese deep learning network for authorship verification. Our model utilizes three LSTM layers followed by a Manhattan distance layer to evaluate the similarity between emails, making effective use of unique stylistic and linguistic characteristics.

Using the SeFACED dataset, our method achieves an accuracy of 90.12%, proving its effectiveness and competitiveness against state-of-the-art methods. This demonstrates the potential of authorship verification as a solution for phishing emails classification.

General conclusion

The rapid digitalization of modern society has ushered in an era of unprecedented interconnectivity, enabling individuals and organizations to access a wealth of information and opportunities. However, this digital transformation has also given rise to a proliferation of cybersecurity threats, with phishing attacks emerging as a pervasive and increasingly sophisticated challenge. Phishing attacks exploit human psychology and technological vulnerabilities, making them one of the most effective and damaging forms of cybercrime. This thesis addresses this critical issue by developing and evaluating advanced techniques for detecting phishing attacks across multiple vectors, including URLs, emails, and SMS messages. Leveraging cutting-edge machine learning and deep learning methodologies, this research makes significant contributions to the field of cybersecurity, offering innovative solutions to mitigate the risks posed by these evolving threats.

The primary objective of this thesis was to develop robust and adaptive solutions for detecting phishing attacks while addressing the unique challenges posed by social engineering tactics. Social engineering attacks, such as phishing, rely on manipulating human behavior rather than exploiting technical vulnerabilities, making them particularly difficult to detect using traditional security measures. To tackle this challenge, we focused on creating models that not only detect phishing attempts but also adapt to the ever-changing tactics employed by attackers. Our work spans multiple dimensions of phishing detection, from analyzing URL structures and email content to verifying authorship and preserving user privacy in SMS-based phishing detection.

One of the key contributions of this research is the use of TCNs for phishing URL detection. By analyzing the sequential patterns of URL characters, this approach achieved 98% accuracy in identifying malicious URLs. TCNs are particularly effective for this task because they can capture long-range dependencies in sequential data, making them well-suited for detecting subtle patterns indicative of phishing attempts. This method provides a proactive tool for identifying and mitigating phishing attempts targeting online resources, helping users avoid malicious websites that could lead to data breaches or financial fraud.

For phishing email detection, we developed a hybrid approach that integrates deep learning with CBR. This method achieved 99.61% accuracy and 98.71% precision, with a low false positive rate, demonstrating its effectiveness in distinguishing phishing emails from legitimate ones. The integration of CBR allows the model to leverage historical attack patterns, enabling it to adapt to new phishing tactics over time. This adaptability is crucial in the fast-evolving landscape of cyber threats, where attackers constantly refine their methods to bypass detection systems. Additionally, our Siamese network-based authorship verification technique achieved 90.12% accuracy in detecting phishing emails by identifying inconsistencies in writing style compared to known legitimate communications. This approach provides an additional layer of defense against social engineering tactics, as it focuses on the unique linguistic patterns of individual authors, making it harder for attackers to impersonate trusted entities.

To address the growing threat of smishing, we developed a privacy-preserving federated deep learning framework. This approach achieved 92.83% accuracy in detecting smishing attacks while ensuring that user data remains private and secure. Federated learning enables collaborative model training across multiple devices without transferring sensitive data to a central server, making it ideal for applications where privacy is a top priority. This innovation paves the way for scalable and personalized phishing detection solutions that respect individual data sovereignty, addressing one of the key challenges in modern cybersecurity.

Furthermore, our work on social engineering attack detection achieved 96% accuracy by addressing class imbalance through oversampling techniques. Class imbalance is a common issue in social engineering detection datasets, where the number of legitimate messages far outweighs the number of social engineering attempts. By using oversampling methods, we ensured that the model could learn from a balanced dataset, improving its ability to detect social engineering attempts without being biased toward the majority class. This approach ensures robust and reliable detection of social engineering attempts across diverse datasets, making it a valuable tool for organizations and individuals alike.

The performance of the proposed models depends heavily on the quality and diversity of the training data. Limited or biased datasets may reduce the generalizability of the results. While our DL-CBR models demonstrated consistent performance across different datasets, challenges remain in fully automating the revision process and refining case retention strategies, which are critical for maintaining adaptability to evolving threats. These areas represent key opportunities for future research.

Additionally, the computational complexity of some models, such as Temporal Convolutional Networks (TCNs) and federated learning frameworks, may pose challenges for real-time deployment in resource-constrained environments. For instance, while TCNs are highly effective for sequential data analysis, they require significant computational resources, which may limit their applicability in low-power devices. Similarly, federated learning, while privacy-preserving, introduces additional overhead due to the need for distributed training and communication between devices.

Looking ahead, this research opens up several promising avenues for future investigation. One key direction is the automation of the case-based reasoning revision process to enhance adaptability to emerging phishing techniques. By automating the incorporation of new attack patterns into the model, we can ensure that it remains effective against evolving threats. Another area of interest is the exploration of semi-supervised and self-supervised learning techniques, which could enable the leveraging of larger volumes of unlabeled data. This would reduce dependence on manual annotation, making the models more scalable and cost-effective. Additionally, the development and evaluation of multilingual datasets, particularly in languages such as Arabic and French, will be crucial in assessing the cross-cultural efficacy of the proposed models. Finally, extensive real-world testing will be necessary to validate the practical applicability and robustness of the developed systems across diverse operational environments, ensuring that they can handle the complexities of real-world cybersecurity challenges.

Bibliography

[Avr, 2008] (2008). *Case-Based Reasoning Approach*, pages 51–70. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning.

[Abutair et al., 2019] Abutair, H., Belghith, A., and AlAhmadi, S. (2019). CBR-PDS: a case-based reasoning phishing detection system. *Journal of Ambient Intelligence and Humanized Computing*, 10(7):2593–2606.

[Alam et al., 2020] Alam, M. N., Sarma, D., Lima, F. F., Saha, I., Ulfath, R.-E., and Hossain, S. (2020). Phishing attacks detection using machine learning approach. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 1173–1179.

[Alawneh et al., 2020] Alawneh, L., Mohsen, B., Al-Zinati, M., Shatnawi, A., and Al-Ayyoub, M. (2020). A comparison of unidirectional and bidirectional lstm networks for human activity recognition. pages 1–6.

[Alhogail and Alsabih, 2021] Alhogail, A. and Alsabih, A. (2021). Applying machine learning and natural language processing to detect phishing email. *Computers & Security*, 110:102414.

[Aljofey et al., 2020] Aljofey, A., Jiang, Q., Qu, Q., Huang, M., and Niyigena, J.-P. (2020). An effective phishing detection model based on character level convolutional neural network from url. *Electronics*, 9(9).

[Alkhalil et al., 2021] Alkhalil, Z., Hewage, C., Nawaf, L., and Khan, I. A. (2021). Phishing attacks: A recent comprehensive study and a new anatomy. 3.

[Alqahtani et al., 2020] Alqahtani, H., Sarker, I., Kalim, A., Hossain, S., Ikhlaq, S., and Hossain, S. (2020). *Cyber Intrusion Detection Using Machine Learning Classification Techniques*, pages 121–131.

[Aouchiche et al., 2024] Aouchiche, R. I. A., Boumahdi, F., Remmide, M. A., and Madani, A. (2024). Authorship attribution in twitter: a comparative study of machine learning and deep learning approaches. *International Journal of Information Technology*, pages 1–8.

[Applegate, 2009] Applegate, S. D. (2009). Social engineering: Hacking the wetware! *Information Security Journal: A Global Perspective*, 18:40 – 46.

[APWG, 2021a] APWG (2021a). Anti-Phishing Working Group. (2021). Phishing Activity Trends Report 4th Quarter 2021. [online] Available at: https://docs.apwg.org/reports/apwg_trends_report_q4_2021.pdf [Accessed 19 March 2024].

[APWG, 2021b] APWG (2021b). Anti-Phishing Working Group. (2021). Phishing Activity Trends Report 3rd Quarter 2021. [online] Available at: https://docs.apwg.org/reports/apwg_trends_report_q3_2021.pdf [Accessed 15 Janv 2022].

[APWG, 2022] APWG (2022). Anti-Phishing Working Group. (2022). Phishing Activity Trends Report 4th Quarter 2022. [online] Available at: https://docs.apwg.org/reports/apwg_trends_report_q4_2022.pdf [Accessed 15 March 2024].

[APWG, 2023] APWG (2023). Anti-Phishing Working Group. (2021). Phishing Activity Trends Report 4th Quarter 2023. [online] Available at: https://docs.apwg.org/reports/apwg_trends_report_q4_2023.pdf?_gl=1*t7dwkj*_ga*MTUxMTAwNDA2Ny4xNzE1MDYyNzYw*_ga_55RF0RHXSR*MTcxNTA2Mjc2MC4xLjEuMTcxNTA2Mjc4OC4wLjAuMA..[Accessed 30 March 2024].

[Aroyo et al., 2018] Aroyo, A. M., Rea, F., Sandini, G., and Sciutti, A. (2018). Trust and social engineering in human robot interaction: Will a robot make you disclose sensitive information, conform to its recommendations or gamble? *IEEE Robotics and Automation Letters*, 3(4):3701–3708.

[Awan, 2020] Awan, M. (2020). Pishing attacks in network security. *LC International Journal of STEM.*

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409.

[Bahnsen et al., 2017] Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., and Gonzalez, F. A. (2017). Classifying phishing urls using recurrent neural networks. *eCrime Researchers Summit, eCrime*, pages 1–8.

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271.*

[Balim and Gunal, 2019] Balim, C. and Gunal, E. S. (2019). Automatic detection of smishing attacks by machine learning methods. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*, pages 1–3. IEEE.

[Barathi Ganesh et al., 2018] Barathi Ganesh, H. B., Vinayakumar, R., Anand Kumar, M., and Soman, K. P. (2018). Distributed representation using target classes: Bag of tricks for security and privacy analytics Amrita-NLP@IWSPA-2018. In *CEUR Workshop Proceedings*, volume 2124, pages 10–15.

[Basit et al., 2021] Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., and Kifayat, K. (2021). A comprehensive survey of ai-enabled phishing attacks detection techniques. *Telecommunication Systems*, 76:139–154.

[Batista et al., 2004] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29.

[Benenson et al., 2017] Benenson, Z., Gassmann, F., and Landwirth, R. (2017). Unpacking spear phishing susceptibility. pages 610–627.

[Bergmann et al., 2005] Bergmann, R., Kolodner, J., and Plaza, E. (2005). Representation in case-based reasoning. *The Knowledge Engineering Review*, 20(3):209–213.

[Bezuidenhout et al., 2010] Bezuidenhout, M., Mouton, F., and Venter, H. S. (2010). Social engineering attack detection model: Seadm. In *2010 Information Security for South Africa*, pages 1–8. IEEE.

[Bhatia et al., 2020] Bhatia, A., Dalton, A., Mather, B., Santhanam, S., Shaikh, S., Zemel, A., Strzalkowski, T., and Dorr, B. J. (2020). Adaptation of a lexical organization for social engineering detection and response generation. *arXiv preprint arXiv:2004.09050*.

[Bhuvana et al., 2021] Bhuvana, Bhat, A., Shetty, T., and Naik, M. (2021). A study on various phishing techniques and recent phishing attacks. *International Journal of Advanced Research in Science, Communication and Technology*, pages 142–148.

[Biswas et al., 2018] Biswas, S. K., Devi, D., and Chakraborty, M. (2018). A hybrid case based reasoning model for classification in internet of things (iot) environment. *J. Organ. End User Comput.*, 30:104–122.

[Boenninghoff et al., 2019] Boenninghoff, B., Nickel, R. M., Zeiler, S., and Kolossa, D. (2019). Similarity learning for authorship verification in social media. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

[Boukari et al., 2021] Boukari, B. E., Ravi, A., and Msahli, M. (2021). Machine learning detection for smishing frauds. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2. IEEE.

[Bountakas et al., 2021] Bountakas, P., Koutroumpouchos, K., and Xenakis, C. (2021). A comparison of natural language processing and machine learning methods for phishing email detection. In *The 16th International Conference on Availability, Reliability and Security*, ARES 2021, New York, NY, USA. Association for Computing Machinery.

[Bountakas and Xenakis, 2023] Bountakas, P. and Xenakis, C. (2023). Helphed: Hybrid ensemble learning phishing email detection. *Journal of Network and Computer Applications*, 210:103545.

[Chen et al., 2023] Chen, S.-S., Sun, C.-Y., and Pai, T.-W. (2023). Using machine learning for efficient smishing detection. In *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, pages 207–208.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

[Cialdini Robert, 2021] Cialdini Robert, B. (2021). Influence, new and expanded: the psychology of persuasion. *City/Country. New York.*

[CLAIR collection of fraudemail, 2008] CLAIR collection of fraudemail (2008). Radev,D.2008‘CLAIR collection of fraudemail , ACL Data and Code Repository , adcr2008t001’, http://aclweb.org/aclwiki[Accessed: 02 March 2020].

[Cyber Threat, 2020] Cyber Threat (2020). Cyber Threat: Report on 2020 Shows Triple- Digit Increases across all Malware Types. [online] Available at: https://www.deepinstinct.com/news/cyber-threat-report-on-2020-shows-triple-digit-increases-across-all-malware-types.

[Dalton et al., 2020] Dalton, A., Aghaei, E., Al-Shaer, E., Bhatia, A., Castillo, E., Cheng, Z., Dhaduvai, S., Duan, Q., Hebenstreit, B., Islam, M. M., Karimi, Y., Masoumzadeh, A., Mather, B., Santhanam, S., Shaikh, S., Zemel, A., Strzalkowski, T., and Dorr, B. J. (2020). Active defense against social engineering: The case for human language technology. In Bhatia, A. and Shaikh, S., editors, *Proceedings for the First International Workshop on Social Threats in Online Conversations: Understanding and Management*, pages 1–8, Marseille, France. European Language Resources Association.

[Das et al., 2024] Das, S., Ahsan, S. M. M., Rahman, M., and Karim, M. S. (2024). A voting approach for heart sounds classification using discrete wavelet transform and cnn architecture. *SN Comput. Sci.*, 5(2).

[DBIR Report, 2021] DBIR Report (2021). verizon. (2021). 2021 Data Breach Investigations Report. [online] Available at: https://www.verizon.com/business/resources/reports/2021-data-breach-investigations-report.pdfx [Accessed 15 March 2022].

[DBIR Report, 2022] DBIR Report (2022). verizon. (2022). 2022 Data Breach Investigations Report 4th Quarter 2022. [online] Available at: [Accessed 23 Jan 2023].

[Derakhshan et al., 2021] Derakhshan, A., Harris, I. G., and Behzadi, M. (2021). Detecting telephone-based social engineering attacks using scam signatures. In *Proceedings of the 2021 ACM Workshop on Security and Privacy Analytics*, IWSPA ’21, page 67–73, New York, NY, USA. Association for Computing Machinery.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Digital GLOBAL OVERVIEW REPORT, 2022] Digital GLOBAL OVERVIEW REPORT (2022). Digital 2022: Global overview report.

[Douzi et al., 2017] Douzi, S., Amar, M., and Ouahidi, B. E. (2017). Advanced phishing filter using autoencoder and denoising autoencoder. *ACM International Conference Proceeding Series*, pages 125–129.

[Edwards et al., 2017] Edwards, M., Larson, R., Green, B., Rashid, A., and Baron, A. (2017). Panning for gold: Automatically analysing online social engineering attack surfaces. *Comput. Secur.*, 69:18–34.

[Egozi and Verma, 2018] Egozi, G. and Verma, R. (2018). Phishing email detection using robust nlp techniques. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 7–12.

[Enron email dataset, 2011] Enron email dataset (2011). Enron email dataset, http://www.cs.cmu.edu/~./enron/[Accessed: 02 March 2022].

[Fang et al., 2019] Fang, Y., Zhang, C., Huang, C., Liu, L., and Yang, Y. (2019). Phishing email detection using improved rcnn model with multilevel vectors and attention mechanism. *IEEE Access*, 7:56329–56340.

[Figueroa et al., 2017] Figueroa, N., L'huillier, G., and Weber, R. (2017). Adversarial classification using signaling games with an application to phishing detection. *Data Min. Knowl. Discov.*, 31(1):92–133.

[Giorgi et al., 2020] Giorgi, G., Saracino, A., and Martinelli, F. (2020). Email spoofing attack detection through an end to end authorship attribution system. In *ICISSP*, pages 64–74.

[Gitelman, 2014] Gitelman, L. (2014). *Paper Knowledge: Toward a Media History of Documents*. Sign, Storage, Transmission. Duke University Press, United States. Includes bibliographical references (pages 189-204) and index.

[Goel and Jain, 2018] Goel, D. and Jain, A. K. (2018). Smishing-classifier: a novel framework for detection of smishing attack in mobile environment. In *Smart and Innovative Trends in Next Generation Computing Technologies: Third International Conference, NGCT 2017, Dehradun, India, October 30-31, 2017, Revised Selected Papers, Part II 3*, pages 502–512. Springer.

[Grace et al., 2016] Grace, K., Maher, M., Wilson, D. C., and Najjar, N. (2016). Combining cbr and deep learning to generate surprising recipe designs. pages 154–169.

[Gupta et al., 2018] Gupta, B. B., Arachchilage, N., and Psannis, K. (2018). Defending against phishing attacks: Taxonomy of methods, current issues and future directions. *Telecommunication Systems*, 67.

[Gupta et al., 2015] Gupta, P., Srinivasan, B., Balasubramaniyan, V., and Ahamad, M. (2015). Phoneypot: Data-driven understanding of telephony threats. In *NDSS*, volume 107, page 108.

[Gupta et al., 2016] Gupta, S., Singhal, A., and Kapoor, A. (2016). A literature survey on social engineering attacks: Phishing attack. *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 537–540.

[Hai and Hwang, 2018] Hai, Q. T. and Hwang, S. O. (2018). Detection of malicious urls based on word vector representation and ngram. *Journal of Intelligent & Fuzzy Systems*, 35(6):5889–5900.

[Halgaš et al., 2020] Halgaš, L., Agrafiotis, I., and Nurse, J. (2020). *Catching the Phish: Detecting Phishing Attacks Using Recurrent Neural Networks (RNNs)*, pages 219–233. Springer International Publishing.

[HB et al., 2018] HB, B. G., R, V., KP, S., and M, A. K. (2018). Distributed representation using target classes: Bag of tricks for security and privacy analytics amrita-nlp@iwspa 2018.

[Heartfield and Loukas, 2018] Heartfield, R. and Loukas, G. (2018). Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework. *Computers & Security*, 76:101–127.

[Hegdal and Kofod-Petersen, 2019] Hegdal, S. S. and Kofod-Petersen, A. (2019). A cbr-ann hybrid for dynamic environments. CEUR Workshop Proceedings.

[Hina et al., 2021] Hina, M., Ali, M., Javed, A. R., Ghabban, F., Khan, L. A., and Jalil, Z. (2021). Sefaced: Semantic-based forensic analysis and classification of e-mail data using deep learning. *IEEE Access*, 9:98398–98411.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hosseinia and Mukherjee, 2018] Hosseinia, M. and Mukherjee, A. (2018). Experiments with neural networks for small and large scale authorship verification.

[Hu et al., 2016] Hu, X., Xia, B., Skitmore, M., and Chen, Q. (2016). The application of case-based reasoning in construction management research: An overview. *Automation in Construction*, 72:65–74.

[Huang et al., 2019] Huang, Y., Yang, Q., Qin, J., and Wen, W. (2019). 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)*, pages 112–119. IEEE.

[Jain and Gupta, 2018] Jain, A. K. and Gupta, B. (2018). Rule-based framework for detection of smishing messages in mobile environment. *Procedia Computer Science*, 125:617–623.

[Jain and Gupta, 2019] Jain, A. K. and Gupta, B. B. (2019). Feature based approach for detection of smishing messages in the mobile environment. *Journal of Information Technology Research (JITR)*, 12(2):17–35.

[Jain et al., 2022a] Jain, A. K., Gupta, B. B., Kaur, K., Bhutani, P., Alhalabi, W., and Almomani, A. (2022a). A content and url analysis-based efficient approach to detect smishing sms in intelligent systems. *International Journal of Intelligent Systems*, 37(12):11117–11141.

[Jain et al., 2022b] Jain, U., Srivastava, Y., Malik, A., Dhingra, D., Kumar, A., and Nagrath, P. (2022b). Malicious dns detection and prediction using smote-enn and hybrid artificial neural network. In *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 138–144.

[Jose nazario phishing email corpus, 2004] Jose nazario phishing email corpus (2004). Jose nazario phishing email corpus, https://monkey.org/~jose/phishing/[Accessed: 02 March 2020].

[Kamruzzaman et al., 2023] Kamruzzaman, A., Thakur, K., Ismat, S., Ali, M. L., Huang, K., and Thakur, H. N. (2023). Social engineering incidents and preventions. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0494–0498.

[Kang et al., 2014] Kang, A., Dong Lee, J., Kang, W. M., Barolli, L., and Park, J. H. (2014). Security considerations for smart phone smishing attacks. In *Advances in Computer Science and its Applications: CSA 2013*, pages 467–473. Springer.

[Karhani et al., 2023] Karhani, H. E., Jamal, R. A., Samra, Y. B., Elhajj, I. H., and Kayssi, A. (2023). Phishing and smishing detection using machine learning. In *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 206–211.

[Krombholz et al., 2015] Krombholz, K., Hobel, H., Huber, M., and Weippl, E. (2015). Advanced social engineering attacks. *Journal of Information Security and Applications*, 22:113–122. Special Issue on Security of Information and Networks.

[Kumar et al., 2020] Kumar, A., Chatterjee, J. M., Díaz, V. G., et al. (2020). A novel hybrid approach of svm combined with nlp and probabilistic neural network for email phishing. *International Journal of Electrical and Computer Engineering*, 10(1):486.

[Lai et al., 2015] Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2267–2273. AAAI Press.

[Lan, 2021] Lan, Y. (2021). Chat-oriented social engineering attack detection using attention-based bi-lstm and cnn. In *2021 2nd International Conference on Computing and Data Science (CDS)*, pages 483–487.

[Lansley et al., 2020] Lansley, M., Mouton, F., Kapetanakis, S., and Polatidis, N. (2020). Seader++: social engineering attack detection in online environments using machine learning. *Journal of Information and Telecommunication*, 4(3):346–362.

[Lara-Benítez et al., 2020] Lara-Benítez, P., Carranza-García, M., Luna-Romera, J. M., and Riquelme, J. C. (2020). Temporal convolutional networks applied to energy-related time series forecasting. *Applied Sciences*, 10(7).

[Le et al., 2018] Le, H., Pham, Q., Sahoo, D., and Hoi, S. C. (2018). Urlnet: Learning a url representation with deep learning for malicious url detection. *arXiv preprint arXiv:1802.03162*.

[Lee et al., 2020] Lee, Y., Saxe, J., and Harang, R. (2020). Catbert: Context-aware tiny bert for detecting social engineering emails. *arXiv preprint arXiv:2010.03484*.

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

[Liang et al., 2022] Liang, Y., Wang, Q., Xiong, K., Zheng, X., Yu, Z., and Zeng, D. (2022). Robust detection of malicious urls with self-paced wide amp; deep learning. *IEEE Transactions on Dependable and Secure Computing*, 19(2):717–730.

[Liu et al., 2021] Liu, M., Zhang, Y., Liu, B., Li, Z., Duan, H., and Sun, D. (2021). Detecting and characterizing sms spearphishing attacks. In *Annual Computer Security Applications Conference*, pages 930–943.

[Lopez and Camargo, 2022] Lopez, J. C. and Camargo, J. E. (2022). Social engineering detection using natural language processing and machine learning. In *2022 5th International Conference on Information and Computer Technologies (ICICT)*, pages 177–181.

[LOPEZ DE MANTARAS et al., 2005] LOPEZ DE MANTARAS, R., MCSHERRY, D., BRIDGE, D., LEAKE, D., SMYTH, B., CRAW, S., FALTINGS, B., MAHER, M. L., COX, M. T., FORBUS, K., and et al. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3):215–240.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation.

[Mambina et al., 2022] Mambina, I. S., Ndibwile, J. D., and Michael, K. F. (2022). Classifying swahili smishing attacks for mobile money users: A machine-learning approach. *IEEE Access*, 10:83061–83074.

[Mashtalyar et al., 2021] Mashtalyar, N., Ntaganzwa, U. N., Santos, T., Hakak, S., and Ray, S. (2021). Social engineering attacks: Recent advances and challenges. In *HCI for Cybersecurity, Privacy and Trust: Third International Conference, HCI-CPT 2021, Held as Part of the 23rd HCI International Conference, HCII 2021, Virtual Event, July 24–29, 2021, Proceedings*, page 417–431, Berlin, Heidelberg. Springer-Verlag.

[Merton Lansley and Polatidis, 2020] Merton Lansley, Francois Mouton, S. K. and Polatidis, N. (2020). Seader++: social engineering attack detection in online environments using machine learning. *Journal of Information and Telecommunication*, 4(3):346–362.

[Mishra and Soni, 2019] Mishra, S. and Soni, D. (2019). A content-based approach for detecting smishing in mobile environment. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India.*

[Mishra and Soni, 2021] Mishra, S. and Soni, D. (2021). Dsmishsms-a system to detect smishing sms. *Neural Computing and Applications*, pages 1–18.

[Mishra and Soni, 2022] Mishra, S. and Soni, D. (2022). Implementation of 'smishing detector': an efficient model for smishing detection using neural network. *SN Computer Science*, 3(3):189.

[Mishra and Soni, 2023] Mishra, S. and Soni, D. (2023). Sms phishing dataset for machine learning and pattern recognition. In Abraham, A., Hanne, T., Gandhi, N., Manghirmalani Mishra, P., Bajaj, A., and Siarry, P., editors, *Proceedings of the 14th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2022)*, pages 597–604, Cham. Springer Nature Switzerland.

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

[Mouton et al., 2014] Mouton, F., Leenen, L., Malan, M. M., and Venter, H. S. (2014). Towards an ontological model defining the social engineering domain. In Kimppa, K., Whitehouse, D., Kuusela, T., and Phahlamohlaka, J., editors, *ICT and Society*, pages 266–279, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Mouton et al., 2015] Mouton, F., Leenen, L., and Venter, H. (2015). Social engineering attack detection model: Seadmv2. In *2015 International Conference on Cyberworlds (CW)*, pages 216–223.

[Mouton et al., 2018] Mouton, F., Nottingham, A., Leenen, L., and Venter, H. (2018). Finite state machine for the social engineering attack detection model: Seadm. *SAIEE Africa Research Journal*, 109(2):133–148.

[Nguyen et al., 2018] Nguyen, M., Nguyen, T., and Nguyen, T. H. (2018). A deep learning model with hierarchical LSTMs and supervised attention for anti-phishing. In *CEUR Workshop Proceedings*, volume 2124, pages 29–38.

[Ozen et al., 2024] Ozen, I., Subramani, K., Vadrevu, P., and Perdisci, R. (2024). Senet: Visual detection of online social engineering attack campaigns.

[Parvinder, 2017] Parvinder (2017). Cyber crimes in india: An overview. *International Journal of Research*, 4:1707–1709.

[Peng et al., 2018] Peng, T., Harris, I., and Sawa, Y. (2018). Detecting phishing attacks using natural language processing and machine learning. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 300–301.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Perner, 2019] Perner, P. (2019). Case-based reasoning – methods, techniques, and applications. In Nyström, I., Hernández Heredia, Y., and Milián Núñez, V., editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 16–30, Cham. Springer International Publishing.

[Pinto et al., 2019] Pinto, T., Faia, R., Navarro-Cáceres, M., Santos, G., Corchado, J., and Vale, Z. (2019). Multi-agent-based cbr recommender system for intelligent energy management in buildings. *IEEE Systems Journal*, 13:1084–1095.

[P.M et al., 2023] P.M, D., M, M., B, N., R.S, S., M.E, P., and A, M. (2023). Identification of phishing attacks using machine learning algorithm. *E3S Web of Conferences*.

[Pous et al., 2011] Pous, C., Pla, A., Gay, P., and López, B. (2011). exit*cbr: A framework for case-based medical diagnosis development and experimentation. *Artificial intelligence in medicine*, 51 2:81–91.

[Radford and Narasimhan, 2018] Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

[Raffel et al., 2019] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

[Rao and Pais, 2017] Rao, R. S. and Pais, A. R. (2017). An enhanced blacklist method to detect phishing websites. In Shyamasundar, R. K., Singh, V., and Vaidya, J., editors, *Information Systems Security*, pages 323–333, Cham. Springer International Publishing.

[Rasymas and Dovydaitis, 2020] Rasymas, T. and Dovydaitis, L. (2020). Detection of phishing urls by using deep learning approach and multiple features combinations. *Baltic journal of modern computing*, 8(3):471–483.

[Remmide et al., 2024a] Remmide, M. A., Boumahdi, F., Ammar Aouchiche, I. R., Guendouz, A., and Boustia, N. (2024a). A robust approach to authorship verification using siamese deep learning: application in phishing email detection. *International Journal of Speech Technology*, pages 1–8.

[Remmide et al., 2022a] Remmide, M. A., Boumahdi, F., and Boustia, N. (2022a). Phishing email detection using bi-gru-cnn model. In Ragab Hassen, H. and Batatia, H., editors, *Proceedings of the International Conference on Applied CyberSecurity (ACS) 2021*, pages 71–77, Cham. Springer International Publishing.

[Remmide et al., 2024b] Remmide, M. A., Boumahdi, F., and Boustia, N. (2024b). Advancing automated social engineering detection with oversampling-based machine learning. *International Journal of Security and Networks*, 19(3):150–158.

[Remmide et al., 2024c] Remmide, M. A., Boumahdi, F., and Boustia, N. (2024c). Towards a hybrid approach combining deep learning and case-based reasoning for phishing email detection. *International Journal on Artificial Intelligence Tools*, 0(ja):null.

[Remmide et al., 2022b] Remmide, M. A., Boumahdi, F., Boustia, N., Feknous, C. L., and Della, R. (2022b). Detection of phishing urls using temporal convolutional network. *Procedia Computer Science*, 212:74–82.

[Remmide et al., 2024d] Remmide, M. A., Boumahdi, F., Ilhem, B., and Boustia, N. (2024d). A privacy-preserving approach for detecting smishing attacks using federated deep learning. *International Journal of Information Technology*, pages 1–7.

[Resnik, 1986] Resnik, A. J. (1986). Book review: Influence: Science & practice.

[Richter and Weber, 2013] Richter, M. and Weber, R. (2013). *Case-based reasoning: a textbook*. Springer.

[Ross et al., 2021] Ross, R. M., Rand, D. G., and Pennycook, G. (2021). Beyond "fake news": Analytic thinking and the detection of false and hyperpartisan news headlines. *Judgment and Decision making*, 16(2):484–504.

[Roy et al., 2020] Roy, P. K., Singh, J. P., and Banerjee, S. (2020). Deep learning to filter sms spam. *Future Generation Computer Systems*, 102:524–533.

[Saha et al., 2020] Saha, I., Sarma, D., Chakma, R. J., Alam, M. N., Sultana, A., and Hossain, S. (2020). Phishing attacks detection using deep learning approach. In *Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020*, number Icssit, pages 1180–1185.

[Sahingoz et al., 2019] Sahingoz, O. K., Buber, E., Demir, O., and Diri, B. (2019). Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357.

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

[Salahdine and Kaabouch, 2019] Salahdine, F. and Kaabouch, N. (2019). Social engineering attacks: A survey. *Future Internet*, 11:89.

[Saleilles and Aïmeur, 2021] Saleilles, J. and Aïmeur, E. (2021). Secubot, a teacher in appearance: How social chatbots can influence people. In *Proceedings of the 1st Workshop on Adverse Impacts and Collateral Effects of Artificial Intelligence Technologies—AIofAI 2021*, volume 2942, pages 31–49. CEUR Montréal, Canada.

[Sawa et al., 2016] Sawa, Y., Bhakta, R., Harris, I. G., and Hadnagy, C. (2016). Detection of social engineering attacks through natural language processing of conversations. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 262–265.

[Scherer et al., 2010] Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

[Seifollahi et al., 2017] Seifollahi, S., Bagirov, A., Layton, R., and Gondal, I. (2017). Optimization based clustering algorithms for authorship analysis of phishing emails. *Neural Processing Letters*, 46:411–425.

[Seth and Damle, 2022] Seth, P. and Damle, M. (2022). A comprehensive study of classification of phishing attacks with its ai/i detection. In *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, pages 370–375.

[Shahrivari et al., 2020] Shahrivari, V., Darabi, M. M., and Izadi, M. (2020). Phishing detection using machine learning techniques.

[Sheikhi et al., 2020] Sheikhi, S., Kheirabadi, M. T., and Bazzazi, A. (2020). An effective model for sms spam detection using content-based features and averaged neural network. *International Journal of Engineering*, 33(2):221–228.

[Sheng et al., 2009] Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., and Zhang, C. (2009). An empirical analysis of phishing blacklists.

[Shravasti and Chavan,] Shravasti, S. S. and Chavan, M. Smishing detection: Using artificial intelligence.

[Shrestha et al., 2017] Shrestha, P., Sierra, S., González, F., Montes, M., Rosso, P., and Solorio, T. (2017). Convolutional neural networks for authorship attribution of short texts. In Lapata, M., Blunsom, P., and Koller, A., editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674, Valencia, Spain. Association for Computational Linguistics.

[Sohn, 2016] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[Sonowal, 2020] Sonowal, G. (2020). Detecting phishing sms based on multiple correlation algorithms. *SN computer science*, 1(6):361.

[Spam assassin project, 2015] Spam assassin project (2015). Spam assassin project (2015) spam assassin public corpus, https://spamassassin.apache.org/publiccorpus/[Accessed: 02 March 2020].

[Tao et al., 2019] Tao, L., Youpeng, H., Wen, Z., and Jie, Z. (2019). The metering automation system based intrusion detection using random forest classifier with smote+enn.

[The National University of Singapore SMS Corpus:, 2023] The National University of Singapore SMS Corpus: (Accessed: 15 March 2023). , https://www.kaggle.com/datasets/rtatman/the-national-university-of-singapore-sms-corpus.

[Tidy, 2020] Tidy, J. (2020). Twitter hack: What went wrong and why it matters.

[Tsinganos et al., 2022a] Tsinganos, N., Fouliras, P., and Mavridis, I. (2022a). Applying bert for early-stage recognition of persistence in chat-based social engineering attacks. *Applied Sciences*, 12(23).

[Tsinganos et al., 2023] Tsinganos, N., Fouliras, P., and Mavridis, I. (2023). Leveraging dialogue state tracking for zero-shot chat-based social engineering attack recognition. *Applied Sciences*, 13(8).

[Tsinganos et al., 2022b] Tsinganos, N., Mavridis, I., and Gritzalis, D. (2022b). Utilizing convolutional neural networks and word embeddings for early-stage recognition of persuasion in chat-based social engineering attacks. *IEEE Access*, 10:108517–108529.

[Tsinganos et al., 2018] Tsinganos, N., Sakellariou, G., Fouliras, P., and Mavridis, I. (2018). Towards an automated recognition system for chat-based social engineering attacks in enterprise environments. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–10.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[Vinayakumar et al., 2018] Vinayakumar, R., Barathi Ganesh, H. B., Anand Kumar, M., Soman, K. P., and Poornachandran, P. (2018). DeepAnti-PhishNet: Applying deep neural networks for phishing email detection CEN-AISecurity@IWSPA-2018. In *CEUR Workshop Proceedings*, volume 2124, pages 39–49.

[Vrbančič et al., 2020] Vrbančič, G., Fister, I., and Podgorelec, V. (2020). Datasets for phishing websites detection. *Data in Brief*, 33:106438.

[Wang et al., 2019] Wang, W., Zhang, F., Luo, X., and Zhang, S. (2019). Pdrcnn: precise phishing detection with recurrent convolutional neural networks. *Security and Communication Networks*, 2019.

[Weerasinghe and Greenstadt, 2020] Weerasinghe, J. and Greenstadt, R. (2020). Feature vector difference based neural network and logistic regression models for authorship verification. In *CEUR workshop proceedings*, volume 2695.

[Wei et al., 2020] Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., and Woźniak, M. (2020). Accurate and fast url phishing detector: A convolutional neural network approach. *Computer Networks*, 178.

[Yan and Cheng, 2024] Yan, A. and Cheng, Z. (2024). A review of the development and future challenges of case-based reasoning. *Applied Sciences*, 14(16).

[Yin et al., 2021] Yin, J., Qi, C., Chen, Q., and Qu, J. (2021). Spatial-spectral network for hyperspectral image classification: A 3-d cnn and bi-lstm framework. *Remote Sensing*, 13(12).

[Zamir et al., 2020] Zamir, A., Khan, H., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A., and Hamdani, M. (2020). Phishing web site detection using diverse machine learning algorithms. *The Electronic Library*, ahead-of-print.

[Zaremba et al., 2014] Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[Zhou et al., 2015] Zhou, C., Sun, C., Liu, Z., and Lau, F. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

[Zhu et al., 2020] Zhu, R., Liao, W., and Wang, Y. (2020). Short-term prediction for wind power based on temporal convolutional network. *Energy Reports*, 6:424–429. 2020 The 7th International Conference on Power and Energy Systems Engineering.