Adversary-Aware, Machine Learning-based Detection of Spam in Twitter Hashtags

Niddal H. Imam

Doctor of Philosophy

University of York

Computer Science

July 2021

ABSTRACT

Concerns about the vulnerability of Machine Learning (ML) to adversarial examples in cybersecurity systems have been growing in recent years. These systems are operating in adversarial environments, so any solutions need to consider the presence of adversaries and to evolve over time in the face of emerging threats. However, most of existing ML-based models designed for cybersecurity systems, such as Online Social Networks (OSNs)' spam detection are either adversary-agnostic models or only focus on one aspect of adversarial environments.

The goal of this work is to design adversary-aware ML-based detectors of spam in Twitter considering three key points: the robustness to adversarial examples, adaptability to evolving attacks and interpertability to security analysts. Throughout the thesis, we used health-related spam campaigns in Twitter Arabic hashtags as a case study. The analysis of these campaigns help us to identify three adversarial attacks and develop three adversary-aware ML- and DL-based detectors. The first contribution of this thesis is a taxonomy of potential adversarial attacks scenarios in Twitter. Then, we moved forward to develop an adversary-aware spam detector, which was built on the observation that the targeted campaigns were found to be using unique hijacked accounts to fool the deployed spam detectors. We designed a new feature, which is faster to compute compared to features used in the literature, and which also improves the accuracy of detecting the identified hijacked accounts by 73%. Additionally, we proposed an approach for designing adversary-aware spam image detectors. The key novelty is that our approach improves the robustness through adversarial training and uses black/ white list with human-in-the-loop (HITL) approach to ensure the detectors can evolve over time. The developed adversary-aware Optical Character Recognition (OCR)-based detector outperforms two SOTA OCRs in recognising Arabic and English text embedded in Twitter spam images. We further propose an OCR post-correction algorithm, which improves the robustness of OCR-based detectors with at least 10% against the generated Adversarial Text Images.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First and foremost, all Praise to ALLAH S.W.T the Almighty, for giving me the blessing, strength to complete my PhD thesis. Also, I would like to thank:

My family, wife Dr. Wjoud Almadani, and two daughters Sama and Deema, for supporting me in the low moments.

Dr. Vasileios Vasilakis and Prof Dimitris Kolovos, for mentoring, supporting, and encouragement.

Saudi Culture of Mission, for its financial support to my PhD scholarship.

DECLARATION

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. All sources are acknowledged as References. The content of some of the chapters has been published in the following papers:

Published.

- **Imam, N.H.** and Vassilakis, V.G., 2019. A Survey of Attacks Against Twitter Spam Detectors in an Adversarial Environment. Robotics, 8(3), p.50.
- **Imam, N.** and Vassilakis, V., 2019, September. Detecting Spam Images with Embedded Arabic Text in Twitter. In 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW) (Vol. 6, pp. 1-6). IEEE
- **Imam, N.** Vassilakis, V., 2019. An Approach for Detecting Image Spam in OSNs. Proceedings of the Conference for Truth and Trust Online 2019.

Under Review

- **Imam, N.** and Vassilakis, V., Kolovos D., Towards Designing An Adversary-Aware Detection of Twitter Spam Campaigns *Journal of Digital Forensics, Security and Law*
- **Imam, N.** and Vassilakis, V., Kolovos D., An Empirical Analysis of Health-Related Campaigns on Twitter Arabic Hashtags *Online Social Networks and Media*
- **Imam, N.** and Vassilakis, V., Kolovos D., OCR post-correction for Detecting Adversarial Text Images, *Information Security and Applications*

Chapter 1

Introduction

Online Social Networks (OSNs), such as Facebook and Twitter, have become a very important part of daily life. People use them to make friends, communicate with each other, read the news, and share their opinions and ideas. A platform such as Twitter is mainly used to inform users about breaking news and trending topics [210]. The amount of information shared in these OSNs has continued to increase in recent years. Studies show that the number of profiles on Facebook, Twitter, and LinkedIn reached more than two billion in 2016 [9]. According to the Statista website [1], Facebook, the most popular OSNs in the world, had nearly 2.8 billion global monthly active users and a total of 3.3 billion users accessing Facebook, WhatsApp, Instagram, and Messenger on a monthly basis cumulatively as of the fourth quarter of 2020. Furthermore, Twitter, which was founded in 2006, has become one of the most popular microblogging platforms [60]. It has been reported that over the course of a month, Twitter has two million users sharing 8.3 million tweets per hour [189]. Additionally, the amount of user activity in OSNs increases considerably during social events, such as sports tournaments, royal weddings, or presidential elections [88].

Twitter's mission statement is, "to serve the public conversation. Violence, harassment and other similar types of behavior discourage people from expressing themselves, and ultimately diminish the value of global public conversation. Our rules are to ensure all people can participate in the public conversation freely and safely." [2]. Unfortunately, even with these rules, the high popularity of Twitter and other OSNs has made them very attractive to malicious users, or spammers. Spammers spread false information, propaganda, rumours, fake news, or unwanted messages [121]. Spam is an unsolicited message that is received from a random sender with no relationship to the receiver. These messages may contain malware, advertisements, or Uniform Resource Locators (URLs) directing the recipients to malicious websites [26]. Although spam is prevalent in all forms of online communication (such as email and the web), researchers' and practitioners' attention has increasingly shifted to spam in OSNs, due to the growing number of spammers and the possible negative effects on users [221, 26]. Some of the characteristics of OSNs have been abused by spammers to spread their spam very quickly, which consequently helps them to reach more users. Many online markets have taken advantage of Twitter hashtags, which are phrases preceded with the "#" symbol, to promote their

[1] https://www.statista.com/topics/751/facebook/
[2] https://help.twitter.com/en/rules-and-policies/twitter-rules

products. Such unsolicited bulk messages are considered to be spam because they could negatively impact users' experience. Dewan and Kumaraguru [88] add that such activities not only affect users' experiences but also violate Facebook's terms of service. Similarly, Twitter's rules [3] state that promoting third-party services or apps to accrue more followers is considered a spam activity. Several reported incidents show the danger of spammers in OSNs. For example, a number of NatWest bank customers fell victim of a phishing attack on Twitter that used spam tweets that looked very similar to those from the official NatWest customer support account [9]. A recent study by Cresci et al. [74] demonstrated that the increase in the number of OSN spammers, who distribute unsolicited spam and advertise untrustworthy products, has an effect on the public's perception of companies, which can eventually lead to people's opinion becoming biased. According to Dewan and Kumaraguru [88], spammers can make $200 million just by posting links on Facebook.

The first appearance of spam on Facebook was in 2008, while the first Twitter spam attack, in which a number of Twitter accounts were hacked to spread advertisements, was in 2009 [245, 277]. On Twitter, spammers tweet for several reasons, such as to spread advertisements, disseminate pornography, spread viruses, phishing, or simply compromise a system's reputation [29]. Furthermore, in [100], the authors asserted that a tweet is considered spam if it is not composed purely of text. Instead, it contains a hashtag, a mention, a URL, or an image. Various types of spam are found on OSNs, including textual pattern spam [291], image spam [37, 36], URL-based spam [268], and phone number-based spam [121]. Whilst most previous studies have focused on detecting the above types of spam, few have attempted to detect advertisement spam. The authors in [220] categorized adversarial advertisements as counterfeit goods, misleading or inaccurate claims, phishing, arbitrage, and malware. The diversity of spam on OSNs makes it very hard for any single existing method to detect most spam [111].

The issue of spamming over OSNs has become an area of interest for many researchers. Many solutions have been proposed to defend against a type of adversaries who try to spread spam massages by using techniques such as blacklisting and whitelisting, or Machine Learning (ML). ML techniques have been shown to be effective when deployed to solve cybersecurity issues in different domains, such as email spam filters, Intrusion Detection Systems (IDSs), and malware detectors [64]. ML models aim to automatically classify messages as either spam or non-spam. These models are categorised into three groups based on the training methods: supervised, unsupervised, and semi-supervised. Various OSNs spam detectors have been developed using ML algorithms; including Support Vector Machine (SVM) [29], Random Forests (RF) [192, 274] and, more recently, Deep Learning (DL) [26]. These ML algorithms are trained to detect spam through analysing messages' content-based (e.g., text, URL, or phone number) or statistic info-based features (e.g., account age, or number of followers) [268, 122]. Also, few studies utilize an Optical Character Recognition (OCR) systems to detect spam images [48]. While these approaches focus on detecting spam messages that targeted users, very limited studies take into account possible adversarial attacks that may be carried out by adversaries to undermine the detection systems (i.e., designing adversary-aware models).

Recently, concerns about the vulnerability of ML models to adversarial examples (i.e., carefully devised input to mislead detection at test-time) [32] are growing. The vulnerability of ML models is not new, and the seminal work in the area of Adversarial Machine Learning was published in 2004,

[3] https://help.twitter.com/en/rules-and-policies/twitter-rules

where Dalvi et al. [83], and later Lowd and Meek [182, 183] studied the problem in the context of email spam filtering. Since then, a large amount of studies has been published, such as developing attacks against ML models at training time (poisoning attacks) and at test time (evasion attacks), including the discovery of adversarial examples against deep networks [247]. However, most of existing ML-based spam detectors focus on the detection task of spam messages without considering the adversarial nature of cybersecurity systems. The arms race between defenders and adversaries in adversarial environments is never ending. Hence, investigating the vulnerability of ML-based spam detection in particular, is very important.

1.1 Motivations

Spam in OSNs can be sent either as a direct message or posted under chat groups or trending hashtags. The detection of spam posted on groups or hashtags has attracted researchers' attention not only because they may irritate users, but also because these messages can be used to distribute more sophisticated security threats, such as malware or ransomware. ML models have been widely adapted to automate the detection of spam in Twitter and other OSNs. However, concerns about the vulnerability of ML models to adversarial examples have been growing recently. The cybersecurity systems are adversarial environments, where arms race between the system designer and the adversaries is never ending, so any designed solution needs to consider the presence of an adversary and to evolve over time in the face of new emerging attacks [223]. The traditional assumption concerning the stationarity of data distribution in ML models is that the dataset used for training and testing share a similar underlying distribution. This assumption is violated in adversarial environments, as adversaries are able to manipulate data, either during training *Causative attack* or before testing *Evasion attack* [231, 6]. Hence, ML models may become the weakest link in a cybersecurity system [86]. Although several studies have investigated the robustness of ML models to adversarial examples in cybersecurity systems, such as IDS, email filtering, and malware detection, it is often overlooked in OSNs' spam detection.

Most of related studies focus on the detection of malicious accounts (e.g., hijacked accounts or bots) that attack users in OSNs without considering the robustness to adversarial examples that might be crafted to undermine or mislead the deployed ML model. Such detectors are vulnerable to different adversarial attacks as they were not designed to function in adversarial environments [226, 7, 40]. Despite the fact that a few recent studies have investigated the robustness of ML models designed for detecting spam in OSNs, several challenges have yet to be solved

One of the challenges when using ML models for spam detection is that the adversarial attacks are non-intrusive; adversaries can access the ML models using the same channel as legitimate users to gain some knowledge [225]. Figure 1.1 demonstrates how an adversary can use the same channel as legitimate users to access the model and learn some of its characteristics. Adversaries can send adversarial examples, which are inputs designed to produce incorrect output, to cause misclassification [52].

Another major problem that ML models must consider when used in adversarial environments is adapting to continuous changes in the capabilities of the adversaries. In other words, how to ensure the ML models can evolve over time in the face of emerging attacks. Not considering the adaptability of the designed ML model may affect the long term performance as the adversaries are constantly

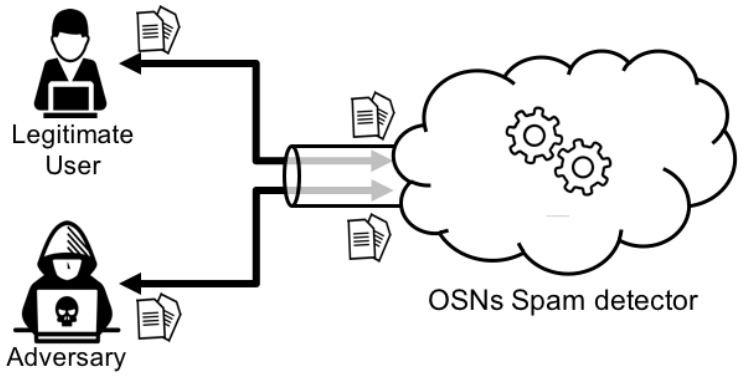


Figure 1.1: An Adversary uses the same channel as legitimate users to exploit knowledge about the detection system

trying to evade detection, which can change the underlying distributions of malicious samples (i.e., *Adversarial concept drift*). Keeping the deployed ML model updated, ensuring that new labelled data are continuously available, which in the case of OSNs is expensive and time-consuming, are challenging tasks when operating in an adversarial setting.

Moreover, another challenge that needs to be considered is the trade off between model interpretability and model accuracy. Modern complex models, such as DL and capsule networks are often yield to better performance, but are perceived as black-box models. It is difficult for a human to understand how these models make their decisions under the hood. The opposite of a black-box model is a white-box model (a.k.a. interpretable model). Debugging ML-based models is crucial when they are used in adversarial environments, where the arms race between adversaries and defenders is never ending. To ensure an ML model designed for cybersecurity systems can effectively thwart evolving attacks, humans (i.e., security analysts) need to be able to know why the predictive performance drops and which part of the system gets affected. Hence, designing a human-interpartable and accurate ML model is important.

The goal of this thesis is to design adversary-aware ML-based detectors of Twitter's spam to tackle the following three challenges. Instead of limiting adversaries access to the deployed ML model, we focus on improving the robustness by detecting identified adversarial examples. Also, to ensure the designed adversary-aware ML-based detectors can evolve over time to face emerging attacks, we focus on the adaptability to possible new adversarial activities such as adversarial drift, and human-interpartable so a security analyst can interact and debug the detectors when it is needed. Since attack scenarios is an application-specific issue [33], we used ongoing spam campaigns that spread healthcare advertisements in Twitter Arabic hashtags as a case study. We designed our adversary-aware ML-based detectors by taking into account three key points: robustness to identified adversarial examples, adaptability to emerging threats, and human-interpretability to debug the detectors.

1.2 Research Questions

Motivated by the aforementioned challenges, this thesis aims to answer the following key questions:

- **RQ1** What are the possible adversarial attacks on Twitter trending hashtags?
- **RQ2** Are existing ML-based detectors of Twitter spam vulnerable to adversarial examples?
- **RQ3** How can we design adversary-aware ML-based detectors of spam in Twitter trending hashtags?

To answer these research questions, we follow the general framework for assessing the performance of ML models in adversarial environments. The framework suggests the following steps:

1. Identifying possible adversarial attacks scenarios against Twitter' spam detectors:
    - *We analyse messages posted by the ongoing spam campaigns in Twitter hashtags to identify possible attack scenarios.*
2. Evaluating the robustness of Twitter' spam detectors to adversarial examples. After analysing the targeted spam campaigns, we identify three adversarial attacks carried out by the targeted spam campaigns in this step:
    - *Spam campaigns use hijacked accounts as adversarial examples to fool the deployed ML-based detectors.*
    - *Spam campaigns inject misspelt spam words into images that can be used as adversarial examples against OCR-based detectors.*
    - *Spam campaigns use images with manipulated embedded text as adversarial examples against OCR-based detectors.*
3. Improving and evaluating the robustness of Twitter's spam detectors to adversarial examples. After evaluating two types of Twitter spam detectors to the identified adversarial examples, we improve their robustness and propose the following detectors:
    - *An adversary-aware ML-based detector of Adversarial Hijacked Accounts*
    - *An adversary-aware OCR-based detector of Adversarial Spam Images*
    - *An adversary-aware OCR-based detector of Adversarial Text Images*

1.3 Thesis Outline and Contributions

Throughout this thesis, we study ongoing spam campaigns that spread untrustworthy healthcare advertisements in Twitter Arabic hashtags. This approach helps us to identify three adversarial attacks on Twitter hashtags and to propose three adversary-aware spam detectors. This thesis contributed to the literature by introducing an approach for designing adversary-aware spam detectors that are robust, adaptable, and interpretable. To summarize, the contributions of this thesis are as follows:

1. **A taxonomy of potential adversarial attacks in Twitter hashtags [142].**

   Problem: Concerns about the vulnerability of ML models to adversarial examples have been increasing recently. Whilst several studies have examined the vulnerability of IDSs, email filters, and malware detectors, few have investigated the vulnerability of OSNs' spam detectors. Since attack scenarios is an application-specific issue [39], identifying potential attacks against Twitter spam detectors is needed. To the best of our knowledge, a taxonomy of possible attacks scenarios against Twitter's spam detectors has not been proposed. Recent studies have suggested that the achievement of a secure system necessitates the prediction of potential attacks (i.e., before they occur) to develop suitable countermeasures.

   Solution: After studying the ongoing health-related spam campaigns in Twitter, we proposed a taxonomy of potential attacks scenarios against Twitter spam detectors [142]. Examples of adversarial activities on Twitter that were discovered after observing Arabic trending hashtags are discussed in detail. A new type of spam tweet (we call them *adversarial spam tweet*), which can be used either to undermine or fool a deployed classifier, is introduced. In addition, possible countermeasures that could increase the robustness of Twitter spam detectors to such attacks are discussed. This research is discussed in Chapter 3.

2. **Spam Campaigns Detection: a robust feature for improving the robustness to adversarial hijacked accounts.**

   Problem: Existing ML-based spam detectors including hijacked accounts detectors tend to be aggressive against tweets that are posted by newly created accounts with a large number of posts, or accounts that suddenly change their behaviours, such as language or retweeting. However, after analyzing ongoing health-related spam campaigns, we found that these campaigns use unique hijacked accounts (we call them Adversarial Hijacked Accounts) as adversarial examples to fool the deployed ML-based spam detectors. Existing approaches for detecting hijacked accounts, which build a behaviour profile for each user, have some limitations, such as they are not applicable for early detection and are computationally expensive that make them not applicable for detecting spam in Twitter hashtags [258].

   Solution: We design a robust feature (*avg_posts*), which can help detecting spam tweets' posted by the adversarial hijacked accounts at a tweet-level in trending hashtags. The new feature leverages accounts' temporal patterns (i.e., account age and number of posts). The effectiveness of the designed feature in the classification task was evaluated and compared with State-Of-The-Art (SOTA) techniques using four methods. First, we measure the importance of the new feature to the deployed classifier using three feature selection and ranking techniques. Then, we evaluate the importance of the proposed feature in detecting the adversarial hijacked accounts using six ML algorithms trained in a supervised manner and three unsupervised autoencoders. The results show that the detection accuracy of most ML algorithms improves when using the new feature by over 10%; the same is true for unsupervised models. Also, two well known datasets were used to evaluate the importance of the new feature. Finally, we develop an adversary-aware ML-based detector, which outperforms SOTA bots and hijacked accounts detectors in detecting the identified adversarial hijacked accounts, while it also incurs less computational cost. This research is discussed in Chapter 4.

3. **Spam Image Detection: improving robustness by fine-tuning with adversarial spam images and a proposed text classification model [139, 140]**.

   Problem: After studying the targeted spam campaigns, a substantial amount of spam image was found. We found that these campaigns inject spam words inside images and purposely misspell embedded spam in images (i.e., Adversarial Spam Image) to fool the deployed OCR-based detector. OCR systems have shown promising results when used for scene text or scanned documents text recognition; however, extracting text from images uploaded into Twitter is challenging due to the complexity of the background and the variability of text regions and rotation. Although some recent studies use an OCR system for detecting spam images in OSNs, very few consider the robustness of these OCR-based detectors in an adversarial setting. These systems become vulnerable to adversarial examples when used in cybersecurity systems (e.g., spam image detectors), where adversaries continuously try to evade detection.

   Solution: To design our adversary-aware OCR-based detector of Twitter spam images we consider two points: improving the text recognition of the adopted OCR system (robustness) and the text classification task (adaptability and human-interpretability). First, we investigated whether the recognition of SOTA OCR is reduced when utilized for detecting images uploaded into Twitter. Then, a transfer learning method was preformed through collecting noisy images from Twitter hashtages and fine-tune the pre-trained OCR system to improve the recognition accuracy of Arabic and English embedded text in spam images. Second, we proposed a text classification model, which consists of a black/white list and human-in-the-loop (HITL) approaches to improve the robustness against images with embedded manipulated spam words (i.e., Adversarial Spam Image) and to ensure the detector can evolve over time by detecting new text manipulations. Our results show that the text recognition part of the model can recognize English text with 57% and Arabic text with 46% accuracy, which outperforms SOTA OCR (Google CloudVision OCR and OCR.space) in detecting images uploaded into Twitter. This research is discussed in Chapter 5.

4. **Spam Image Detection: an OCR post-correction algorithm for improving the robustness to adversarial text images**.

   Problem: In this research, we investigate the robustness of OCR-based detectors to the generated Adversarial Text Images that might be carried out by the targeted spam campaigns in Twitter hashtags. In this attack, we assume that adversaries would manipulate embedded text in images by flipping a few characters of malicious or sensitive words with visually similar symbols or numbers to mislead OCR-based detectors. Components of automated OCR-based detectors are rarely checked by a human, which makes them vulnerable to such adversarial attacks. Adversaries can take advantage of this vulnerability to attack the text recognition and text classification parts of OCR-based detectors.

   Solution: To develop our adversary-aware OCR-based detector, we proposed an OCR post-correction algorithm [4] to improve the robustness and Multiple Classifiers System (MCS)-based text classification model to ensure the detector can evolve over time. Results showed that our

[4] https://github.com/niddal-imam/Post-OCR-Correction

proposed algorithm, which uses a spelling checker to de-noise and classify text, improves the robustness of three SOTA OCR models with at least 10% against adversarial text images, and it outperforms five spellcheckers in correcting two types of adversarial text. Also, we evaluated the perception of the generated adversarial text images to human, and the experiments showed that 91% of the participants were able to correctly recognise the adversarial text images. Additionally, we evaluate the developed adversary-aware OCR-based detector using MCS-based text classification model, and showed considerable improvement in the performance. This research is discussed in Chapter 6.

1.4 Thesis Statement

Twitter hashtags are adversarial environments, so any solution needs to consider the presence of an adversary and evolving attacks. Thus, we design adversary-aware detectors of Twitter spam by taking into account the robustness to adversarial examples, the adaptability to evolving attacks, and interpretability to experts.

1.5 Summary of Research Contributions

This section highlights research contributions of this thesis. Different datasets, algorithms, and detectors have been developed and published to ensure experiments reproducibility.
Create new datasets:

- We create datasets of health-related spam campaigns in Twitter trending hashtags for building ML-based spam detectors [134].
- We create synthetic and real-world image datasets with embedded Arabic and English text to build a text recognition part spam image detectors [135].
- We create images datasets collected from Twitter Arabic hashtags for building a text localization part of spam image detectors [136].
- We build synthetic datasets of images with embedded adversarial text to improve the robustness of OCR-based spam detectors [137].

Identify new attacks:

- We identify a new type of hijacked accounts used in health-related spam campaigns as adversarial examples to fool ML-based detection systems.
- We identify a new adversarial attack in Twitter, where adversaries create spam images containing manipulated embedded text to cause image spam detectors to miss-recognise the images' content.

Propose new algorithm:

- We design a robust feature *avg_posts* to detect the newly identified adversarial attack on Twitter trending hashtags. The new feature was designed based on the following observation: the identified hijacked accounts are old accounts that have a very few number of posts in their lifetime.
- We propose an OCR post-correction algorithm for de-noising and classifying adversarial embedded text in images. Specifically, the proposed method has been designed to improve the robustness of OCR-based detectors of images and with embedded adversarial text in OSNs.

Develop new spam detectors:

- We develop an adversary-aware ML-based detector of spam posted by health-related spam campaigns in Twitter hashtags.
- We develop an adversary-aware OCR-based detector that uses a deep learning approach for extracting text from spam images, and a black/ white list approach with human assistance is applied for classifying extracted words [141].
- We develop an adversary-aware OCR-based detector for detecting images with embedded malicious content in Twitter.

1.6 Thesis Structure

This section provides an overview of the chapters of this thesis.

Chapter 1 outlines motivation for approaches and models proposed in this thesis, the research aims and questions, the contributions and the publications.

Chapter 2 presents the background for the research undertaken in this thesis and discuses briefly a critical review of the literature in related fields. It focuses on three fields of study: Twitter spam detection, machine learning, and adversarial machine learning.

Chapter 3 represents the first contribution, and answers the first research question of this thesis. It discuses potential attacks scenarios in Twitter hashtags and possible countermeasures. A taxonomy of adversarial attacks against Twitter spam detectors using common frameworks for evaluating the vulnerability of machine learning models was proposed.

Chapter 4 represents the second contribution of the thesis. The chapter answers the second and third research questions by proposing an adversary-aware ML-based detector. It provides an empirical analysis of health-related spam campaigns and explains how the targeted spam campaigns can fool Twitter detectors by using adversarial hijacked accounts. The proposed adversary-aware spam detector was evaluated and compared with SOTA classifiers following evaluation methods used in related studies.

Chapter 5 represents the third contribution of this thesis, which is adversary-aware OCR-based detector. This chapter answers the second research question as it shows how can we design and robustify image spam detectors that also can evolve over time. It describes the methodologies and experiments used in developing the spam image detectors. Finally, the results of the detector were compared with SOTA OCRs.

Chapter 6 represents the 4th contribution of the thesis. Itdescribes the proposed defence method against images with manipulated embedded text (i.e., adversarial text images). In this chapter, we answer the second and third research questions by proposing an algorithm to improve the robustness and adaptability of OCR-based spam detectors. First, human perceptibility of the generated adversarial text images was evaluated by conducting a user study. Then, the proposed OCR post-correction algorithm was evaluated following three evaluation methods used in related studies.

Chapter 7 concludes the thesis by summarizing the contributions and discussing limitations and future work.

Chapter 2

Background and Related Works

This thesis fits into the crossroads of three research areas: spam detection, machine learning, and adversarial machine learning. To contextualize this thesis, a review of the background information required to understand the thesis and a review of current literature related to the questions and objectives are presented in this chapter. It consists of two sections; Section 2.1 provides background to the researches in this theses, and Section 2.2 covers the relevant literature to address the research goals and questions.

2.1 Background

This section presents a review of the background information to the research, namely ML-based detection, vulnerability, adaptability, and interpretability of ML-based detection of spam in Twitter. Subsection 2.1.1 introduces a brief background on online social networks. Subsection 2.1.2 outlines general steps of building ML models for detecting spam in Twitter. Subsection 2.1.3 presents preliminaries of deep learning and OCR systems. In Subsection 2.1.4, we provide a summary of seminal works that propose frameworks for evaluating the performance of ML models in adversarial environments. In Subsection 2.1.5, we give a brief overview of recent studies that investigate the adaptability of ML-based detectors in adversarial environments. Subsection 2.1.6, provides an overview of ML models' interpretability. This subsection has been added since we consider the interpretability of ML models as an important factor for designing adversarial-aware ML models.

2.1.1 Online Social Networks

The growth and popularity of Web 2.0 applications have created a new way for people around the world to collaborate and communicate. The widespread diffusion of high-speed Internet has led to the emergence of new generation Web 2.0 applications OSNs [67]. Online social network platforms, such as Facebook, Twitter, MySpace, LinkedIn and others provide space for users to create profiles and connect with other's profiles to form social networks. Nowadays, OSNs are key platforms for content and opinion dissemination, professional social networking, recommendations, and political campaigns. One of the main objectives of these platforms is social interaction and connection. Hence, these platforms share a lot of similar futures. For examples Facebook allows user to create

chat groups and similarly users in Twitter can create a hashtag for discussion. However, the fast growing in popularity of these platforms attract spammers who spread false information, propaganda, rumours, fake news, or unwanted messages. Spammers abuse some of the features provided in these platforms to reach more victims in a short time.

Although this thesis focus on the detection of spam in Twitter, there is no practical reasons that could prevent the proposed approach from being applicable in other OSNs. Twitter is chosen because of the following reasons:

- Twitter Streaming Application Programming Interference (API)[1], which allows researchers to extract public tweets.
- Datasets from Twitter are available on a relatively reasonable scale compared to other OSNs.
- Lack of studies that investigate spam campaigns in Arabic trending hashtags.

2.1.2 ML-based Detectors of Twitter Spam

Twitter and the research community have proposed a number of spam detectors to protect users. Twitter uses different methods to fight spammers. It allows users to report spam accounts or suspending accounts that violate its rules. Since these methods can easily be avoided, researchers have been motivated to propose more powerful methods. Generally, approaches for detecting Twitter spam can be divided into automated approaches, including machine learning, and non-automated approaches that require human interaction [274]. In this these, we focus on studying ML-based approaches.

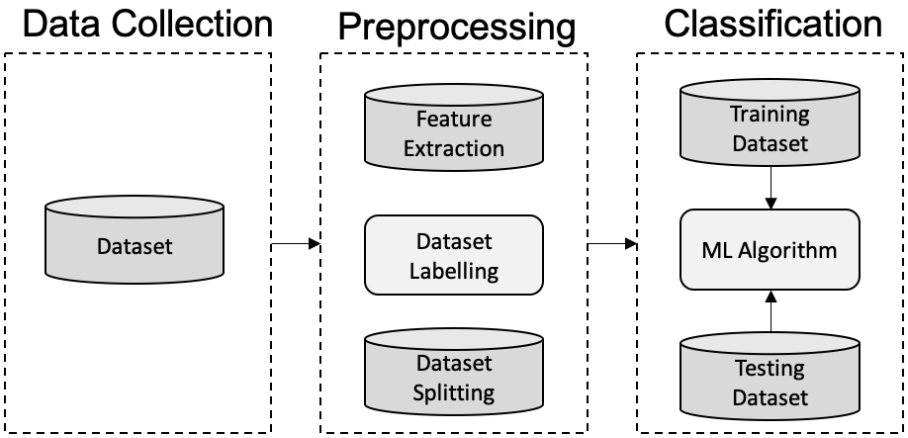


Figure 2.1: Process of building ML-based spam detection

The process of building spam detection using ML comprises three main steps. Figure 2.1 presents the steps of building an ML-based spam detection, which consists of data collaction, preprocessing, and classification. The first step involves collecting data from Twitter using its API. This is followed by preprocessing, which includes feature extraction, dataset labelling, and dataset splitting.

[1]https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

However, for textual spam detectors, the preprocessing step may include more functions, such as tokenizing, removing stop words, and steaming. Extracting and selecting features from tweets or Twitter accounts helps the chosen ML algorithm to distinguish between spam and non-spam tweet. Examples of these features include account age, the number of followers or friends, and the number of characters. Dataset labelling or ground truth is the process in which the collected dataset are labelled either manually or using a crowdsourcing site. The dataset then needs to be split into a training set and a test set. The last step entails training the chosen classification algorithm by using the labeled or unlabeled training dataset, followed by performance evaluation using the testing dataset, after which the trained ML algorithm can be used for spam detection [4, 63].

After labelling or annotating collected data, an important step in the preprocessing is feature extraction. On the basis of surveys in [150, 165], Twitter spam detectors can be classified into four categories based on the type of extracted features: user-based, content-based, hybrid-based, and relation-based techniques. User-based techniques are also referred to as account-based classify tweets according to an account's features and other attributes that provide useful information about users' behaviour. Content-based techniques use a tweet's content, such as the linguistic properties of the text or the number of hashtags in the tweet, for classification. Hybrid-based techniques use a combination of user-based and content-based features. This category is usually used to detect spam in real-time, in contrast to user-based techniques, which can only detect spam after a message has been received. Relation-based techniques can detect a tweet immediately if it is received from an unknown sender. The features used in relation-based techniques are distance and connectivity. Table 2.1 presents features categories, names, and brief descriptions [150, 175].

Table 2.1: Feature categories, names and descriptions

Category	Name	Description
Account-based	account_age	The number of days since the creation of an account
	no_followers	The number of followers of an account
	no_friends	The number of friends an account has
	no_favourites	The number of favourites an account has received
	no_lists	The number of lists an account is a member of
	no_reputation	The ratio of the number of followers to the sum of followers & friends
	no_status	The number of tweets an account has
Content-based	no_mentions	The number of mentions in a tweet
	no_words	The number of words in a tweet
	no_chars	The number of characters in a tweet
	no_hashs	The number of hashtags in a tweet
	no_urls	The number of URLs in a tweet
	no_phone	The number of phone numbers in a tweet
Relation-based	Distance	The length of the distance between accounts
	Connectivity	The strength of the relationship between accounts

Additionally, the authors in [275] categorized the methodologies used for detecting Twitter spam into three groups: syntax-based, feature-based, and blacklist-based detection (see Figure 2.2). Syntax-based detectors analyze a tweet's content, including linguistic features and shortened URLs,

to determine whether the tweet is spam or non-spam. The second group, feature-based detectors, extract a set of statistical features from tweets to help the utilized classifier determine whether the tweet is spam or non-spam. This group uses a combination of techniques: account-based features, tweet-based features, and social graph features. Account-based features include account age and number of followers, while tweet-based features are the number of characters and the number of URLs. However, these types of features can easily be fabricated, so some studies (e.g., Song et al, 2011; Yang et al, 2013) have found that robustness can be increased by adopting a social graph to detect spam by analyzing mathematical features, such as social distance and connectivity between followers as cited in [63]. In blacklist-based detectors, accounts and tweets are blocked on the basis of users' feedback or the URL's reputation. The first study of the effectiveness of some Twitter spam detection techniques that have been used in the past was presented in [119]. Examples included spam behaviour, clickthrough, and blacklists. The authors found that the blacklist methods (for example, Google SafeBrowsing) are too slow at detecting new threats. They found that although 90% of victims visit spam URLs within the first 2 days of receipt, it took 4–20 days for the URLs in spam tweets to be blacklisted. In another study, it was determined that blacklists can protect only a few users, and the authors asserted that studying the regional response rate could improve spam detection. Furthermore, to overcome the limitations of the blacklist, some preliminary studies have used heuristic rules to filter Twitter spam [63].

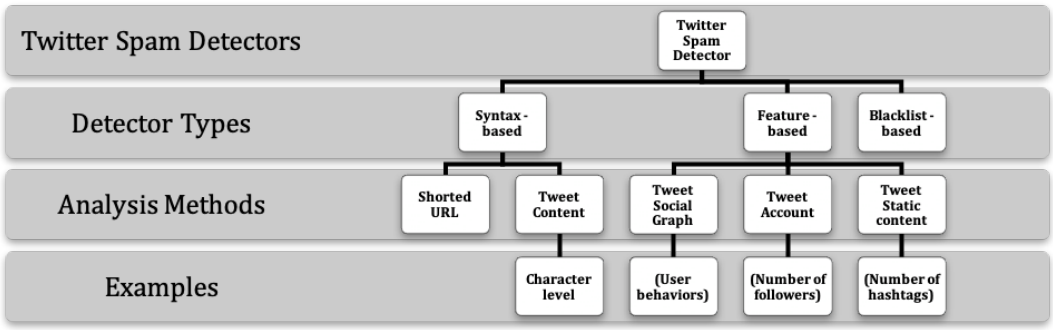


Figure 2.2: Types of twitter spam detectors

The last step in the building process of ML-based spam detection is classification. There are different method for building a classification model that can be categorized into taxonomy, based on the required outcome of the classifier [275] (see Table 2.2). The most common learning techniques are:

1. **Supervised Learning** is the technique where an ML algorithm trained to maps input to the desired outputs. Also, it refers to the task, in which an ML algorithm learned from labeled data. Supervised learning is widely used in the literature for building a spam detection that can classify data into two classes spam or non-spam.

2. **Unsupervised Learning** is the technique that uses unlabeled data for training an ML algorithm to fined relationships between instances. An advantage of this technique, which is more applicable than supervised learning in some cases, is that it doesn't require labeled data. There are two approaches for unsupervised learning: the goal of the first approach is to make

decisions that improve tasks, whereas the goal of the second approach is to fined similarities in the training set.

3. **Semi-Supervised Learning** refers to a technique that uses a few labeled instances and a large number of unlabeled instances for training an ML algorithm, and it has shown promises in detecting spam [71, 16].

Table 2.2: Types of ML techniques

Method	Attributes
Supervised Learning	Learning from a set of labeled data Requires labeled for training data Most common form of learning
Unsupervised Learning	Learning from a set of unlabeled data Finds unseen relationships in the data Most common form is clustering
Semi-supervised Learning	Learning from labeled and unlabeled data Only requires a small set of labeled data Preferable for cases where vast amounts of unlabeled data exist

Table 2.3 shows common types of algorithms used in supervised learning, which deals more with classification [16]. Also, Figures 2.3 and 2.4 represent some supervised and unsupervised algorithms.

Table 2.3: Supervised learning algorithms

Categories	Algorithms
Linear Classifiers	Logical Regression
	Naïve Bayes Classifier
	Perceptron
	Support Vector Machine
Quadratic Classifiers	
K-Means Clustering	
Boosting	
Decision Tree	Random Forest
Neural networks	

2.1.3 OCR-based Detectors of Twitter Spam

In this subsection, we introduce some preliminaries of deep learning and optical character recognition since a umber of recent studies utilized these technique to detect spam images.

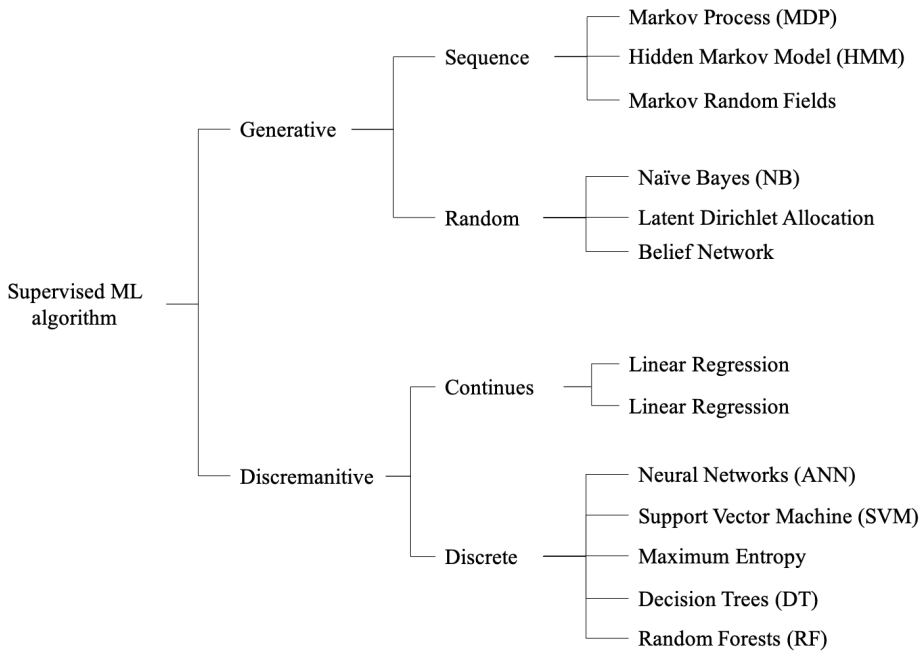


Figure 2.3: Supervised learning algorithms

2.1.3.1 Deep Learning

The aim of deep learning methods is to build powerful learning architectures through stacking simpler entities, such as deep neural networks, convolutional neural networks, recurrent neural networks, or generative adversarial networks [272, 252]. Deep learning has become very popular for many computer vision and image recognition tasks. A deep neural network, is the simplest form of deep network, which is a function $f_\theta : X \to Y$, where θ is a parameter of the function, X and Y are the input and the output space respectively. In images classification tasks, X is a vector space (e.g., images of the same size) and Y is a discrete set of classes (e.g., the set of possible objects in the images). The model f_θ is trained in a supervised learning method to find the best set of parameters θ using the labeled training dataset $D = f\{(x_i; y_i)\}_i$, and the loss metric $L(f(x_i); y_i)$, which measures the difference between the model's prediction $f(x_i)$ and the correct label y_i [240].

2.1.3.2 Optical character recognition

Optical character recognition (OCR) is a technology that extracts digital text from images of handwritten or printed text, such as a scanned document, a page of magazine, or even a photo of a scene that includes signs with texts [240]. The OCR systems generally involves two steps: text detection and recognition. In the first step, text regions in the spam image are detected. Second, the detected text regions are recognized and saved as text file. Figure 2.5 illustrates the architecture of the overall model.

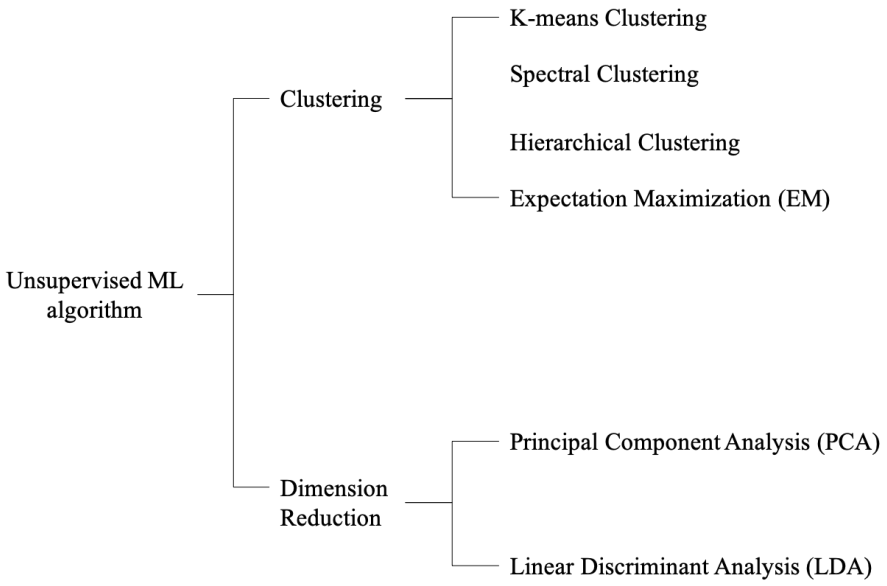


Figure 2.4: Unupervised learning algorithms

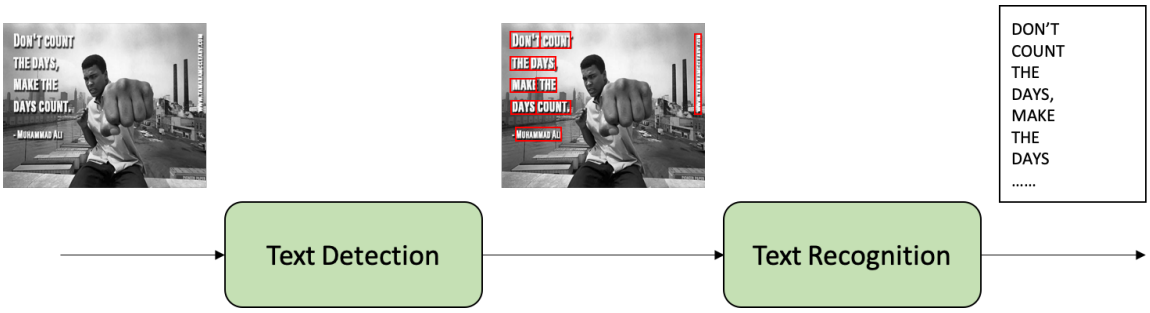


Figure 2.5: Overall architecture of an OCR system.

2.1.3.3 Text Detection Model

Text localisation/detection is the first step in the OCR system, and its objective is to capture word level bounding boxes or segmentation maps [53]. The design of features to distinguish text from backgrounds is the core of text localisation. Traditional methods such as [102, 204, 215] capture the properties of scene text through designing the features manually. However, these features are learned directly from training in deep learning based models. SOTA text localisation such as CTPN [253], TextBoxes [174], SegLink [233], EAST [289] and PixelLink [87] are built on a fully convolutional network [169]. These deep learning or conventional based models consist of several stages, including FCN and Non-maximum suppression (NMS). The FCN [179] is a CNN that consists of fully connected conventional layers; it takes an input image of any size, and produces an output of corresponding spatial dimensions. In FCN, each layer of data is a three-dimensional array of size h x w x d, where h and w are spatial dimensions, and d is the channel dimension. The first layer is the input image, with pixel size h x w, and d color channels. Each memory location in higher convolution layers

correspond to its *receptive* field (i.e., the location or region in the image) unlike fully-connected layers, in which neuron is affected by the entire image.

The segmentation task, in which an image is assigned pixel-wise labels, is usually preformed by adapting FCN. There are two types of segmentation: semantic segmentation, which only consider object category, and instance segmentation that consider both object category and a differentiation of instances. Deng et al., [87] categorized segmentation-based text detection methods into three groups. In semantic segmentation, the detection task is preformed by predicting three kinds of score maps: text/non-text, character classes, and character linking orientations. Then these score maps are grouped into words or lines. The second group is regression-based Text Detection. Most text detectors, such as [253] [289] [174] take advantage of the anchor idea in object detection to predict cropped text. Bounding boxes are obtained from location regression in regression-based methods. However, in the third category (i.e., Instance Segmentation-based), the predicted positive pixels are linked together into text instances, and then the bounding boxes are directly extracted from the instances [87].

2.1.3.4 Text Recognition Model

The objective of text recognition is to take an input image and predict a label distribution y_t for each frame x_t at time t in the feature sequence $x = x_1, ..., x_T$ where T is the sequence length. CNN is not appropriate for sequence-like objects, since it requires the input images to be scaled to a fixed size. Hence, a Convolutional Recurrent Neural Network (CRNN), which is the combination of Deep Convolutional Neural Network (DCNN) and Recurrent Neural Network (RNN) is proposed [234]. The architecture of CRNN consists of three components: Convolutional Layers, Recurrent Layers, and Transcription Layer.

At the bottom of the CRNN, the convolutional layers (CNN without fully connected layers) are constructed to extract sequential features from an input image. The input image is divided into columns from left to right. Each feature map column corresponds to a receptive field of input image. Thus, each vector feature of a feature sequence is considered as a descriptor of that receptive field.

Then, the recurrent network is built at the top of convolutional layers to make predictions for the output, which is a set of feature sequence frames. A label distribution y_t for each frame x_t is predicted by the recurrent layers. One of the advantages of recurrent layers is that RNN can capture contextual information within a feature sequence. This is very helpful as some characters might require several sequential frames to be fully described. However, the traditional RNN suffers from a vanishing gradient problem limits the storage memory and makes the training process time-consuming. Thus, Long-Short Term Memory (LSTM) was used to overcome the vanishing gradient problem. LSTM is a type of RNN, and it consists of memory cells to store the past context, and input/output gates, to store context for a long period of time, and forget gates, which are used to clear the memory cell. A deep bi-directional (BiLSTM), which consists of forward and backward LSTMs, is more suitable for image-based sequences to capture contexts from both directions. Then, at the top of CRNN , the transcript layer translates the pre-frame predictions of the recurrent layers into a label sequence. It finds the label sequence with the highest probability conditioned among the pre-frame predictions. There are two modes of transcription: lexicon-free and lexicon-

based transcriptions. In lexicon-based transcriptions, the prediction of a set of label sequences is constraint to dictionary (e.g. a spellchecking), while the predictions are made without any lexicon in lexicon-free mode. Two options are widely used in this layer for the conditional probability task: (1) Connectionist temporal classification (CTC) [117] and (2) attention-based sequence prediction (Attn) [236]. The key difference between the two is that CTC allows predicting a non-fixed number of a sequence from an input of a fixed number of the features. Whereas, Attn automatically predicts the output sequence from the information flow within the input sequence [17]. CTC provides an alignment-free method for training an end-to-end neural network for sequence labelling tasks.

2.1.3.5 Connectionist temporal classification

In CTC, the probability is defined for label sequence l conditioned on the per-frame predictions $y = y_1, ..., y_T$, where T is the sequence length. Here, each $y_t \in [0,1]^{|\Gamma|}$ is a probability distribution over all alphabet as well as a 'blank' Γ at position t. A sequence-to-sequence mapping function $\mathcal{B}$: $\Gamma^T \rightarrow \Gamma^{\leq T}$, where $\Gamma^{\leq T}$ is the set of sequences of length less than or equal to T over the original label alphabet Γ. $\mathcal{B}$ maps π onto l by removing blank and repeated labels. For example, $\mathcal{B}(c - cat-) = \mathcal{B}(-cc - -aat-) = cat$. Then, the conditional probability is defined as the sum of probabilities of all π that are mapped by $\mathcal{B}$ onto l [234]:

$$p(l|y) = \sum_{\pi:\mathcal{B}(\pi)=l} p(\pi|y) \tag{2.1}$$

2.1.3.6 Attention-based sequence prediction (Attn)

Attention Network is a type of recurrent neural network RNN that directly generates the target sequence $y = y_1, ..., , y_T$ from an input image x. Formally, $Encoder(x) = (h_1, ..., h_T)$. The architecture of attention-based decoder, which first proposed in [18], is that at the $t - th$ step, the decoder generates an output y_t by 2.2 [66]:

$$y_t = generate(s_t, g_t) \tag{2.2}$$

where s_t, which is an RNN hidden state at time t, is computed by Eq 2.3 [66]:

$$s_t = RNN(y_{t-1}, g_t, s_{t-1}) \tag{2.3}$$

and g_t is the weighted sum of sequential feature vectors $(h_1, ..., h_T)$, and computed by 2.4 [66]:

$$g_t = \sum_{j=1}^{T} \alpha_{t,j} h_j \tag{2.4}$$

where $\alpha_t \in \mathbb{R}_T$ is a vector of attention weights (i.e., alignment factors). α_t is often evaluated by scoring each element in $(h_1, ..., h_T)$ separately and then normalizing the scores using Eq 2.5 and 2.6 [18]

$$e_{t,j} = v^T tanh(W_{st-1} + Vh_j + b), \tag{2.5}$$

$$\alpha_{t,j} = \frac{exp\,(e_{t,j})}{\sum_{j=1}^{T} exp\,(e_{t,j})} \tag{2.6}$$

where $v, W, V and\ b$ are all trainable parameters.

Here, the functions in Eq 2.2 represents a feed-forward network, while Eq 2.3 represents a LSTM recurrent network. The attention-based decoder generates sequences of variable lengths by adding a special end-of-sentence (EOS) [246] token to the target set. Thus, the attention-based decode completes the generation of characters when EOS is emitted. The loss function of the attention-based decoder is formulated as follows 2.7 [66]:

$$\mathcal{L}_{Att} = -\sum_{t} inP\,(\hat{y}_t|x, \theta) \tag{2.7}$$

where $\hat{y}_t$ is the ground truth of the $t-th$ character and θ is a vector that combines all the network parameters.

2.1.4 Vulnerability of of ML-based Detectors in Twitter

Despite the success and high accuracy of existing ML-based models in detecting Twitter spam, they may nevertheless vulnerable if they were not developed for adversarial settings. The robustness of ML-based models against adversarial attacks, has recently become subject to increased interest in the research community [45]. Although ML-based models have been widely used for cybersecurity systems, these methods are vulnerable in an adversarial environment. The very first work in the field of Adversarial Machine Learning dates back to 2004, when a group of researchers Dalvi et al., [82] developed a framework and algorithms to detect adversarial activities. Since then, a large number of studies have examined the vulnerability of ML models by developing frameworks for evaluating algorithms [21, 40], launching attacks against ML models, and designing countermeasures [227]. A popular framework for evaluating secure learning was proposed in [23] and extended in [21, 41, 129]; it enables different attack scenarios to be envisaged against ML algorithms. The framework suggests the following steps: (1) to identify potential attacks against ML models by using the popular taxonomy; (2) to simulate these attacks to evaluate the resilience of ML models, and to assume that the adversary's attacks are implemented according to their goals, knowledge, and capabilities/resources; (3) to investigate some possible defense strategies against these attacks. Defending against adversarial attacks is challenging because these attacks are non-intrusive in nature and are launched by an adversary using the same channel as legitimate users. Thus, it is deficult to employ traditional encryption/security techniques as defense strategies against these attacks [225]. Figure 1.1 demonstrates how an adversary can use the same channel as legitimate users to access an ML model and learn some of its characteristics. Designing proactive models rather than traditional reactive models is a necessity in the adversarial environment. Whereas reacting to detected attacks will never prevent future attacks, proactively anticipating adversaries' activities enables the development of suitable defense methods before an attack occurs [41]. This has motivated researchers to formulate different attack scenarios against machine learning algorithms and classification models and propose some countermeasures.

As the spam detection techniques are improving, adversaries attack methods are evolving as well. This arms race has lead to a newly emerging type of adversaries that intend to attack these automated cybersecurity systems not only to evade detection but also to degrade their performance. Considering the detection of spam in an adversarial setting is often overlooked. Most of related studies develop their detection systems from an adversarial agnostic point of view, in which they assume that detecting spam in OSNs can be done by collecting data and training an ML algorithm. However, this naive approach can be violated as a spammer is constantly trying to evade the deployed detector [228]. Adversarial examples are inputs to ML that are designed to produce incorrect outputs [52]. The term was first introduced in [248] and used for computer vision, but in the context of spam and malware detection, the term evasion attacks is used in [41]. Detecting adversarial examples in Twitter and other OSNs is an open issue that requires more investigation.

2.1.5 Adaptability of ML-based Detectors in Twitter.

Twitter hashtags are specific topics used to allow communities and discussions to grow around. Users can initiate a discussion with others by using the hashtag symbol (#) to identify a topic. Once a hashtag starts to be discussed and retweeted by users, it becomes a trend in a given region [190]. The streaming nature of these hashtags makes the detection of spam more challenging due to the fast flow of messages, and changes in the data distribution over time [228, 153]. Also, Twitter trending hashtags are adversarial environments, in which adversaries not only try to avoid detection, but they may attack the deployed detection system to degrade its performance. Recent works have shown that the detection accuracy of ML algorithms decreases over time in a streaming data environment due to changes in the underlying data distribution (*i.e., concept drift*) [61, 63]. Several existing studies [61, 275, 222] addressed the traditional concept drift that occurs due to spamming activities in Twitter. For example, adversaries may change their spamming tactics to evade detection by using phone number or URLs. On the other hand, adversaries may attack the deployed ML-based detector by launching a distributed denial-of-service (DDoS) attack to affect the detector performance which cause (*i.e., adversarial drift*). Sethi and Kantardzic [226] stated that although the adversarial drift phenomenon has been studied in the literature since 2009, they are the first to directly address the problem of adversarial drift in streaming data. Also, the authors stated that in adversarial drift, adversaries causes the drift intentionally to degrade the deployed classifier and to avoid detection by the traditional concept drift detectors. Whereas, the traditional concept drift in OSNs may occur a result of an evolution in people's preferences, population changes (e.g. during special events), the complexity of the environment, or any hidden context [110, 160].

From a probabilistic perspective, an input, or feature vector is donated as x and class label as y. Data stream could be defined as an infinite sequence of (x_i, y_i). Training set D and test set T are two sets of sequentially adjacent examples drawn from the data stream. The labels in T are not known during classification process and will only be provided after some period of time. The existing assumption of data stationary distribution is stated as follows [112]:

$$P_{(x,y)} = P_{(x|y)}.P_{(x)} = P_{(y|x)}P_{(y)} \tag{2.8}$$

However, the difference between data stream mining and traditional mining tasks is that the

data distribution of training and testing data evolves over time in three different ways: (1) feature changes, i.e., the changes in the data distribution $P_{(x)}$; (2) conditional changes, i.e., the changes in the classifier boundary $P_{(y|x)}$ or in the class conditional probability $P_{(x|y)}$; and (3) dual changes, i.e., the changes in both $P_{(x)}$ and $P_{(x|y)}$. Thus, concept drift can be formalized as if the prior probability changes over time. [101, 224]

$$P_{t+1(x,y)} \not\equiv P_{t(x,y)} \tag{2.9}$$

Adversarial drift is a particular type of concept drift and can be distinguished by the following characteristics:

1. The drift is a result of changes to the malicious class samples only.
2. The drift is a result of adversarial activities (e.g. reverse engineering).
3. The drift is always targeted towards subverting the deployed classifier [226].

Adversarial drift occurs as a result of exploratory attacks, which attack instances and cause drift in instances' labels that the classifier must follow. On the other hand, causative attacks cause a shift in the distribution of training data, which eventually leads to classification errors [146]. Consequently, designing an ML-based model for detecting spam in streaming and adversarial environments, such as Twitter hashtags, necessities considering the adaptability to possible drift or evolving attacks.

2.1.6 Interpretability of ML-based Detectors in Twitter.

Detecting spam has becoming a more complex task because as detection techniques improving, adversaries attacking methods are evolving. For efficient detection of spam messages in OSNs and Twitter in particulate, we need to analysis different features of the messages, such as statistical, textual, images and videos features. Adversaries tend to manipulate different types of features to mimic legitimate users messages' or accounts' features. Consequently, a recent trend in spam detection is to use complex ML-based models, for example using an ensemble of classifiers and a Linear Discriminant Analysis (LDA) [68], MCS with a soft max function [149], multiple auto-encoders [259], or MCS with a fully-connected neural network layer [258]. Although these complex detectors prove to achieve better performances, they are often perceived as black-box models. The opposite of black-box models are white-box models also known as human-interpretable models. Although there is not a formal consensus definition for interpretability, it can be referred to "methods and models that make the behaviours and predictions of machine learning systems understandable to humans" [198]. Interpretability and expainability are growing fields in ML, and this increased interest has resulted in many works to discuss different aspects of interpretable/explainable ML from medical diagnosis to decision making in the justice and education systems. Although the two terms interpretability and expainability are often used interchangeably in the literate, there are a few differences. Interpretability enables humans to predict *what is going to happen*, whereas expainability enables humans to explain *what is happening* [106]. The ability to interpret an ML model enables decision makers (e.g., experts or non-experts) to debug, update, and ultimately trust it [164]. Additionally, ML

interpertability can be either global or local. While global interpretation methods are useful for understanding the general mechanisms or debugging a model, local interpretation methods explain individual predictions [198].

Designing interpretable ML models is challenging because interpretability and accuracy are two concepts competing each other. Simplicity and generalization are the main concerns of interpretability, wheres accuracy favoring nuance and exception [164]. There are two approaches that widely used for interpretations of ML models: regression analysis and rule-based ML [198]. These models provide not only predictions, but also descriptions of a class, which are reasons for a prediction [273]. In one hand, linear regression models can be interpreted through analyzing the model structure or a weighted sum of features. For example the weights can be interpreted as the effects that the features have on the prediction. On the other hand, rule-based ML models, such as decision trees [158] or decision sets [273], interpret a learned structure (e.g., IF-THEN) to understand how the model makes predictions. Some recent studies attempt to make complex models interpretable, such as visualizing CNN's features [208], or random forest's feature importance [55]. However, in high-dimensional scenarios, linear regression, decision trees, or complex models may become not interpretable [198]. Additionally, a decision set type of rule-based ML has some shortcomings. Understanding all of the possible conditions that must be satisfied is difficult and limiting the interpretability. Specially in multi-class classification [164].

Although ML models' interpretability has been considered in high-risk applications, such as healthcare [177, 2], finance [123], and fairness [91], it has not been well studied in the context ML spam detection. Since cybersecurity systems are dynamic environment, where adversaries constantly trying to pass through, debugging is crucial. Methods of ML models' interpretability can be used for knowledge discovering, debugging, undrestanding the model's predictions, and controlling and improving the model [198]. According to Molnar [198], ML models can only be debugged or updated if it is interpretable. In contrast to existing spam detectors, this thesis posits that for designing an ML-based spam detection for adversarial environments, the human-interpretability need to be consider.

Designing a simplified approximation model to make complex models interpretable to security analysts is one of the goals of this thesis. We investigate using a rule-based classifier to aggregate complex models (e.g., a DL, or RF) outputs, improve the performance, enable security analysts to interact and debug, and to ensure the detector can evolve over time. As we use the rule-based classifier for aggregation, it consists of a small number of hard-coded IF-THEN rules. Applying our approach for a debugging task is very simple. If a sub-model mispredicted a test point, we can identify and investigate that part of the detector.

2.2 Related Works

This section briefly discusses related works (more detailed literature reviews are provided in the following chapters). It lists the related works to this thesis in detecting spam campaigns in Twitter hashtags (Subsection 2.1.2), detecting spam images in Twitter hashtags (Subsection 2.2.2), and robustifying ML-based spam detectors (Subsection 2.2.3).

2.2.1 Detecting Spam Campaigns in Twitter

A 'spam campaign' refers to a collection of accounts controlled by a 'spammer' to spread malicious content in OSNs [69]. Detecting spam campaigns is a complex task as these campaigns usually use different techniques to mimic legitimate user accounts [10]. Diffierent methods have been proposed for detecting spam campagins, such as designing a robust feature [274], using a collective detection approach [69], or detecting accounts that contain the same phone numbers [122]. Also, spam campaigns use spambots to generate a bulk of spam tweets and spread misinformation. According to [75], social spambots are a growing phenomenon, and current spam detectors designed to detect a single spam account are not capable of capturing spambots. Although their study showed that neither humans nor existing machine learning models could detect spambots accurately, the result of an emerging technique deploying digital DNA has achieved a very promising detection performance. Similarly, the authors in [242] stated that methods designed to detect spam using account-based features cannot detect crowdturfing accounts (accounts created by crowdsourcing sites that have crowdsourcing and astroturfing characteristics). Another study [274] noted that spammers tend to create account bots to quickly reach their goals by systematically posting a large amount of spam in a short period of time. Consequently, the authors proposed an approach that uses the time property (for example, the account creation date and tweet posting time), which cannot be modified by spammers, to reduce the creation of bots. In [111], an approach called *Tangram*, which uses a template-based model to detect spam on OSNs, was proposed. After analyzing the textual pattern of a large collection of spam, the researchers found that the largest proportion of spam was generated with an underlying template compared with other spam categories (for example, paraphrase or no-content).

Additionally, spam campaigns uses compromised accounts (also known as hijacked accounts), which can be considered as *Trolls*. Hijacked accounts are legitimate users' accounts that a malicious party takes control over to spread misinformation or gain financial profit [232]. Existing works on OSN account hijacking can be divided into two categories: analysis and detection. The first category focuses on presenting methods for hijacking, studying hijacked account behaviours, or analyzing the experience of hijacked account users. A research paper that studied 13 million hijacked Twitter accounts [250] categorized the techniques used for compromising accounts on OSNs into four techniques: (1) Database Dumps, where stealing one of a victim's account credentials (e.g., email) can expose their OSN accounts, as users tend to reuse passwords across multiple accounts; (2) Password Guessing, where attackers use brute-force guessing to infer weak passwords; (3) Social Contagion, in which the attackers use social engineering type attacks (e.g., phishing); and, (4) External Contagion, which involves social engineering attacks originating from external platforms. The authors analysis also shows that criminals use compromised accounts to spread weight loss advertisements, to gain followers, or to disseminate 'how to make money' advertisements. On the other hand, the approaches used in the literature for detecting hijacked accounts can be categorized into three groups: textual-based, feature-based, and behavioural-based approaches.

In contrast to the aforementioned studies addressed spam campaigns including bots and hijacked accounts, our research focus on detecting massages posted by these malicious accounts in Twitter hashtags. Detecting spam in trending hashtags requires designing a tweet-level detection approach,

whereas most related studies have adopted user-level approaches. Also, most of the related works rely on analysing messages' content, such as URLs or phone numbers or analysing the relationships between users, which are expensive and time consuming. Furthermore, these studies were developed to detect spam campaigns without considering the presence of adversaries that may attack the detection systems.

2.2.2 Detecting Spam Images in Twitter

Over the last few years, the volume of spam image in OSNs has increased exponentially, which presents challenges for most current ML-based spam detectors. One of the techniques used for image analysis is OCR, a technology that converts images with textual content (for example a handwritten scanned document, or a photo that includes printed text) into digital text. There are two types of OCR models: character-based, which recognize each character, and end-to-end, which recognize the entire sequence of characters [240]. OCR techniques have been widely used for text extraction from document images, but when applied to natural images, these techniques face different challenges, such as the presence of background objects, inconsistent lighting, and a large number of fonts, style, and languages [144].

Detecting spam image in OSNs remains an open issue that required more research. A recent study by Borisyuk et al. [48] developed an OCR-based model that detects and recognizes text in images uploaded to Facebook. The system, called *Rosetta*, consists of two models: text detection and text recognition models. The developed models have implemented in Detectron, open-source software used for object detection research. Also, [282] developed a model called *Malena*, which can detect different types of spam including images carrying text, number, or Quick Response (QR) code and images contain pornography in Chinese social networks. The authors launched different text manipulation attacks to evaluate the text recognition part of the OCR-based model. Tramèr et al. [255] developed a framework for blocking adversarial ads in Facebook and web pages in general. The proposed framework takes a screenshot of the page, and then extracts text from images using Tesseract OCR. The authors evaluate the robustness of their OCR-based model to evasion type of adversarial attacks.

Although few studies investigate designing OCR systems for detecting spam images in OSNs, the vulnerability of these systems requires more research. The OCR-based model *Rosetta* [48] has not been designed for adversarial settings. The other studies [282, 255] have only focus on detecting spam images and the robustness of their models against adversarial examples, but have not considered the adaptability to evolving attacks.

2.2.3 Robustifying ML-based Detectors of Twitter Spam

Adversarial attacks that can be carried out by an adversary against ML models are categorized into two types, which are commonly referred to in the literature as *exploratory* attacks and *causative* attacks. In an exploratory attack, an adversary manipulates a deployed classifier during the testing phase by carefully crafting samples that can bypass the classifier. In contrast, in a causative attack, the adversary attacks during the training phase by contaminating the training data to mislead the deployed classifier [21, 40, 8, 206]. Although defence against adversarial attacks is challenging,

some defence strategies are commonly proposed in the literature, for example randomization [8] and disinformation [231] against exploratory attacks, and data sanitization [54] and robustness [23] against causative attacks.

Few recent studies investigate adversarial attacks against traditional ML-based detector in OSNs. In [269], the authors evaluated the security of an ML detector that is designed to detect spam generated by malicious crowdsourcing users of Weibo (the Chinese version of Twitter) against evasion and poisoning attacks. Their focus was on adversaries that use crowdsourcing sites to launch attacks. To study evasion attacks, two attacks were simulated: basic evasion, in which an adversary has limited knowledge, and optimal evasion, in which the adversary has perfect knowledge. The results showed that an optimal evasion attack has a much higher impact than a basic one. However, in the real world, it is very difficult for adversaries to have perfect knowledge about the system. Thus, the less knowledge that adversaries have about the system, the harder it is for them to evade detection. In causative attacks, two mechanisms for launching poisoning attacks are used. The aim of the first poisoning attack is to mislead the system by using crowdturfing admins to inject misleading samples directly into the training data. In the second poisoning attack, adversaries pollute training data by crafting samples that mimic benign users' behavior. After analyzing both attacks, it was found that injecting misleading samples causes the system to produce more errors than the second poisoning attack.

Another study by [206] analyzed the robustness to evasion and poisoning attacks of a Twitter spam detector called POISED. The detector is designed to distinguish between spam and non-spam messages on the basis of the propagation of messages in each campaign. In a poisoning attack, the goal of an adversary is to contaminate training data by joining communities to alter their network and structure. The adversary posts manipulated messages in these compromised communities to mislead the system. Similarly, in an evasion attack, the adversary joins communities and imitates the propagation of non-spam messages to evade detection. The results showed that, in both attacks, the performance of POISED decreased when the percentage of compromised communities increased. Thus, the authors suggested that an adversary can only successfully attack systems if he or she has perfect knowledge about the structure and network of the targeted community.

Previous works have concluded that an adversary's level of knowledge about the deployed model plays an important role in determining the success of attacks. This supports the common defense strategies used in the literature, namely, disinformation and randomization. Furthermore, the authors in [269] suggested that even if an adversary knows the importance of features used by the deployed model, he or she will not be able to evade detection without knowing the ML algorithm used. However, this cannot stop determined adversaries from trying every way possible to accomplish their goals [231]. Furthermore, as stated in [6], relying on obscurity in an adversarial environment is not good security practice, as one should always overestimate rather than underestimate the adversary's capabilities. Both works concluded that poisoning attacks are more dangerous since models need to be updated over time. Most importantly, both disinformation and randomization approaches focus on making the models hard to attack, but they do not provide measures to be taken once an attack is detected.

2.3 Summary

This thesis differs from most of the existing researches in that its objective is to design adversary-aware ML-based detectors that are robust, adaptable and interpretable. We suggest that these three factors are important when developing an ML-based spam detector in adversarial environment (e.g., Twitter). Most of the approaches proposed in the literature for detecting spam in OSNs focus on detecting a specific form of spam (e.g. spambots) using traditional ML-based approaches, where an ML algorithm is trained in a supervised or unsupervised fashion on data collected from OSNs and then evaluating the model using testing data. On the other hand, a few recent studies use off-the-shelf DL-based models to develop OCR-based models for detecting an spam images. However, most of these studies focus only on improving the detection accuracy of spam messages without taking into account the precnce of an active adversary that might attack the detection system and how these systems can evolve over time. Although few studies investigate the robustness of spam detectors to adversarial examples, the adaptability to evolving attacks and human-interpretability have not been considered.

Relevant works to the researches in this thesis can be divided into three folds: spam campaigns detection in Twitter, OCR-based models for detecting spam images in Twitter, and adversarial attacks against ML-based spam detectors. Table 2.4 presents a comparison between related research that studied the detection of spam campaigns, spam images, and adversarial attacks in OSNs.

Table 2.4: A Comparison between related researches in terms of the three key factors

Title	Methodology	Robustness	Adaptability	Interpretability	Detection Approach
Man vs. Machine- Practical Adversarial Detection of Malicious Crowdsourcing Workers [269]	Different ML algorithms (SVM, RF, and DTs) were evaluated against Evasion and Poisoning attacks.	Yes	No	No	meta feature-based
POISED: Spotting twitter spam off the beaten paths [206]	User-level detection using NLP and ML algorithms (Naive Bayes, SVM and Random Forest). Evasion and Poisoning attacks were performed.	Yes	No	No	Content feature-based (text)
Handling adversarial concept drift in streaming data [229]	A framework for detecting adversarial drift. Robustness against evasion attacks was evaluated using an ML algorithm (Linear SVM)	Yes	Yes	No	meta feature-based
Rosetta: Large Scale System for Text Detection and Recognition in Images [49]	OCR-based model uses Faster-RCNN and CNN for detecting spam images.	No	No	No	Content feature-based (image)
Stealthy Porn: Understanding Real-World Adversarial Images for Illicit Online Promotion [282]	OCR-based model uses PixelLink and Mask R-CNN for detecting spam images. Evasion using noise and text manipulation.	Yes	No	No	Content feature-based (image and QR codes)
AdVersarial: Perceptual Ad Blocking meets Adversarial Machine Learning [255]	OCR-based model uses Tesseract OCR and Yolov3 for detecting spam images. Four evasion attacks against different part of the framework were performed.	Yes	No	No	Content feature-based (image)
COMPA: Detecting Compromised Accounts on Social Networks [95]	Two classifiers: Mandatory consists of time of the day, source, proximity and language, Optional includes links, direct interaction, and topic	No	No	No	Content and meta feature-based
End-to-End Compromised Account Detection [149]	End-to-end framework E2ECAD consists of three components – Temporal Feature Extraction, User Context, and Network Feature Extraction. The results of the components are concatenated and the softmax function is utilized for the final prediction.	No	No	No	Content and meta feature-based
CADET: A Multi-View Learning Framework for Compromised Account Detection on Twitter [259]	Unsupervised framework consists of four auto-encoders. The framework encoded each feature of users' accounts independently using four views (i.e. encoders). Finally, a multi-view decoder is used to aggregate the reconstruction errors of the encoders.	No	No	No	Content and meta feature-based
You have been CAUTE! Early Detection of Compromised Accounts on Social Media [258]	Compromised Account User Tweet Encoder CAUTE consists of two encoders: tweet2user and user2tweet. The encoders transform lexical and meta features of a tweet into more informative and less noisy features. Res2class a fully-connected neural network layer uses the residual errors of the encoders to predict if the post is compromised or not.	No	No	No	Content and meta feature-based

Chapter 3

Identifying Potential Adversarial Attacks in Twitter Hashtags

This chapter investigates possible adversarial attacks scenarios against Twitter spam detectors in trending hashtags. Although a general taxonomy of possible attacks against ML models has been proposed before, we believe that identifying adversarial attacks in Twitter is needed as attack scenarios are an application-specific issue. Adversaries goals, capabilities, and knowledge are not the same in all cybersecurity systems. Thus after absorbing health-related spam campaigns in Arabic trending hashtags, we present different attack scenarios against Twitter spam detectors and discuss potential defence methods. (This chapter includes and expands on my paper previously published in *N. H. Imam and V. G. Vassilakis. A survey of attacks against twitter spam detectors in an adversarial environment.Robotics, 8(3):50, 2019.*)

3.1 Introduction

The high popularity of OSNs, such as Facebook and Twitter and a large amount of data been shared on these platforms have made them very attractive to adversaries, who are spamming users for malicious purposes. Studies show that the number of profiles on Facebook, Twitter, and LinkedIn reached more than 2 billion in 2016 [5]. OSNs' spam can be conveyed in multiple forms (e.g., textual or multimedia) unlike other domains, where images are blocked by default, such as some E-mail systems. ML techniques have proven to provide an automation solution for detecting these spamming activities.

However, as the spam detection techniques are improving, adversaries attacking methods are evolving as well. This arms race has lead to a newly emerging type of adversaries that intend to attack these automated systems not only to evade detection but also to degrade their performance. Although this type of adversaries has been studied in several cybersecurity systems, such as IDS, email filters, and malware detection, few have investigated these adversaries in OSNs. Not considering these adversarial activities at the design stage makes ML models vulnerable to different adversarial attacks either during training phase *Causative attacks* or the prediction phase *Exploratory attacks*. Therefore, investigating the vulnerability of ML-based spam detectors in OSNs

to adversarial examples, along with the design of suitable countermeasures, which are the tasks of Adversarial Machine Learning, are important.

Studying the robustness of OSNs' spam detectors to adversarial attacks is crucial. Robustifying ML models is a very active area of research. To the best of the researcher's knowledge, a taxonomy of possible adversarial attacks against Twitter' spam detectors has not been performed. Recent studies have suggested that the achievement of a secure system necessitates the prediction of potential attacks (i.e., before they occur) to develop suitable countermeasures [42]. Thus, the main goal of this chapter is to present a comprehensive overview of different possible attacks scenarios, which is the first step toward evaluating the robustness of Twitter' spam detectors to adversarial examples. The key contributions of this part of the thesis are threefold.

1. After observing Arabic trending hashtags, it was found that there were very active spam campaigns spreading advertisements for untrustworthy drugs targeting Arabic-speaking users. These campaigns were studied and examples of a new type of spam tweet, which we called *adversarial spam tweet* that can be used by an adversary to attack Twitter spam detectors, are presented.

2. A general taxonomy of the possible adversarial attacks against Twitter' spam detectors is provided. In this research, we propose different attack scenarios in Twitter hashtags using common frameworks for devising attacks against ML models.

3. In addition, potential defense mechanisms that could reduce the effect of such attacks are investigated. Ideas proposed in the literature are generalized to identify potential adversarial attacks and countermeasures. Twitter, which is one of the most popular OSN platforms, is used as a case study, and it is the source of all examples of attacks reported herein.

The remainder of this chapter is structured as follows: Section 3.2 provides an overview of adversarial machine learning. Section 3.3 surveys the adversarial attacks that could be used against Twitter spam detectors and presents a proposed taxonomy of such attacks. The summary of this chapter is discussed in Section 3.4

3.2 Related Works

This section briefly introduces the background and related works in adversarial machine learning. It discusses different adversarial attacks and countermeasures.

3.2.1 Taxonomy of Attacks Against ML

A popular taxonomy proposed in [41, 22, 23] categorizes attacks against ML models along the three following axes:
The attack INFLUENCE

- **Causative**: The attack influences the training data to cause misclassification.

- **Exploratory**: The attack exploits knowledge about the deployed classifier to cause misclassifications without influencing training data.

The type of SECURITY VIOLATION

- **Integrity violation**: An adversary evades detection without compromising normal system operations.
- **Availability violation**: An adversary compromises the normal system functionalities available to legitimate users.
- **Privacy violation**: An adversary obtains private information about the system (such as its users, data, or characteristics) by reverse-engineering the learning algorithm.

The attack SPECIFICITY

- **Targeted** attacks focus on a particular instance.
- **Indiscriminate** attacks encompass a wide range of instances.

The first axis, which is the attack influence, divides an adversary's capability of influencing a classifier's learning systems into causative and exploratory. The influence is causative if an adversary misleads the deployed classifier by contaminating (poisoning) the training dataset through injecting it with carefully crafted samples. In contrast, the influence is exploratory if an adversary gains knowledge about the deployed classifier to cause misclassification at the testing phase without influencing training data.

The second axis describes the type of security violation committed by an adversary. The security violation can be regarded as an integrity violation if it enables an adversary to bypass the deployed classifier as a false negative. In addition, the attack can violate the model's availability if it creates denial of service, in which it misclassifies non-spam samples as spam (false positives), or if it prevents legitimate users from accessing the system. The security violation can be regarded as a privacy violation if it allows an adversary to exploit confidential information from the deployed classifier.

The third axis of the taxonomy refers to the specificity of an attack. In other words, it indicates how specific an adversary's goal is. The attack specificity can be either targeted or indiscriminate, depending on whether the attack (1) causes the classifier to misclassify a single or few instances or (2) undermines the classifier's performance on a larger set of instances.

3.2.2 Common Types of Threat Models

After presenting the taxonomy of attacks against ML models, the next step toward identifying potential attack scenarios is threat modeling, which involves defining an adversary's goal, knowledge, and capability [41, 22, 23]. According to the above taxonomy, the attacker's goal may be based on the type of security violation (integrity, availability, or privacy) and on the attack specificity (targeted or indiscriminate). For instance, the adversary's goal could be to violate the ML models' integrity by manipulating either a specific instance or different instances. An attacker's level of knowledge about the classifier varies and may include perfect knowledge (white-box setting), limited knowledge (gray-box setting), or zero knowledge (black-box setting). Attacker capability can involve either influencing training data (causative attack) or testing data (exploratory attack).

3.2.3 Adversarial Attacks and Defense Strategies

The existing literature on adversarial ML models provides different attack examples and defense methods for both adversarial attack types (causative and exploratory). This section reviews common attack examples and some defense strategies against these attacks (see Table 3.1).

Table 3.1: Common adversarial attacks and defenses

	Causative Attack	Exploratory Attack
Attack	Poisoning Red Herring Label-Flipping	Probing Evasion Reverse Engineering Good Words Attack
Defense	RONI Game Theory based Multiple Learners	Randomization Disinformation

3.2.3.1 Causative Attacks

One of the most common types of causative attack is a poisoning attack, in which an adversary contaminates the training dataset to cause misclassification [21]. An adversary can poison training data by either directly injecting malicious samples or sending a large number of malicious samples to be used by the defender when retraining the model [269]. A label-flipping attack is another example of a causative attack. Here, an adversary flips the label of some samples and then injects these manipulated samples into the training data. Different methods are used to perform this attack. Adversaries can either select samples that are nearest to or farthest from a classifier's decision boundary and flip their label [163]. The easiest method is to randomly flip the label of some samples that might be used for retraining. In [44], it was shown that randomly flipping about 40% of the training data's labels decreased the prediction accuracy of the deployed classifier. A red herring attack is a type of causative attack in which the adversary adds irrelevant patterns or features to the training data to mislead the classifier so that it focuses on these irrelevant patterns [6, 44]. Defending against causative attacks is challenging because ML classifiers need to be retrained periodically to adapt to new changes. Retraining the classifier makes it vulnerable because the data used for retraining are collected from an adversarial environment [163].

3.2.3.2 Causative Defense Methods

Although preventing these attacks is difficult, there are some defense methods proposed in the literature that can reduce the effect of these attacks. Defense methods against causative attacks may rely on Game Theory; in these methods, the defense problem is modeled as a game between the adversary and the classifier [6, 51, 82]. Data sanitization methods focus on removing contaminated samples that have been injected by an adversary from a training dataset before training a classifier, while robust learning focuses on increasing the robustness of a learning algorithm to reduce the influence of contaminated samples [57]. Reject-on-negative-impact (RONI) is one of the simplest and

most effective defense methods against causative attacks and is considered to be a data sanitization method. In RONI, all the training data go through preliminary screening to find and reject samples that have a negative impact on the classification system. To distinguish between contaminated and untainted samples, a classifier is trained using base training data before adding suspicious samples to the base training data and training another classifier. The prediction accuracy for both classifiers on labeled test data is evaluated. If adding suspicious samples to the training data reduces the prediction accuracy, these samples must be removed [129]. Another defense method involves using Multiple Classifiers System (MCS), which has been shown to reduce the influence of poisoned samples in training data [35].

3.2.3.3 Exploratory Attacks

The most popular types of exploratory attacks are evasion and reverse engineering. Both attacks start with a probing attack, in which an adversary sends messages to reveal some information about the targeted classifier. Once the adversary gains some knowledge about the system, he or she can either carefully craft samples that can evade the system (an evasion attack) or use that information to build a substitute system (a reverse-engineering attack) [24]. Furthermore, a Good Word Attack is a type of exploratory attack in which the adversary either adds or appends words to spam messages to evade detection. Good Word attacks can be passive or active. In a passive attack, the adversary constructs spam messages by guessing which words are more likely to be bad or good (for example, a dictionary attack). In an active attack, the adversary has access to a targeted system that enables him or her to discover bad and good words [184].

3.2.3.4 Exploratory Defense Methods

As with causative attacks, it is difficult to prevent exploratory attacks because, in most cases, systems cannot differentiate between messages sent for a legitimate purpose and those sent to exploit the system. However, there are currently two common defense methods: disinformation and randomization. In disinformation methods, the defender's goal is to hide some of the system's functions (for example, concealing the classification algorithms or features used by the classifier) from an adversary. In contrast, in randomization methods, the defender's aim is to randomize the system's feedback to mislead an adversary [24].

Although most of these attack strategies and defense methods were proposed for domains such as email spam filtering, IDS, and malware detection, the underlying approach can be applied to Twitter spam detectors. The following section applies some of these techniques in the context of Twitter spam detectors.

3.3 Our Method

This section describes the methodology used for developing our taxonomy of adversarial attack scenarios in Twitter. It presents possible attacks against Twitter spam detectors. Examples of adversarial spam tweets that can be used by adversaries to attack Twitter are also provided.

3.3.1 The Proposed Taxonomy of Adversarial Attacks in Twitter

Different attack scenarios against Twitter spam detectors are proposed. Attack tactics were defined using the framework of the popular attack taxonomy presented in [24, 21] that categorizes attacks along three axes: influence, security violations, and specificity. This framework was extended in [39] to derive the corresponding optimal attack strategy by modeling an adversary's goal, knowledge, and capability. The adversary's goals considered in this study are either to influence training or test data or to violate the system's integrity, availability, or privacy. The adversary's knowledge is considered to be perfect knowledge (white-box setting) and zero-knowledge (black-box setting). This ensures that both the worst-case and best-case scenarios are considered for an adversary when they attack spam detectors. The adversary's capability is based on their desired goals. For example, if the goal is to influence the training data, the adversary must be capable of doing so. Examples of adversarial spam tweets were extracted from Arabic trending hashtags. The number of spam tweets using Arabic trending hashtags was found to be high, the reasons for which are beyond the scope of this study. However, it was found that there were very active spam campaigns spreading advertisements for untrustworthy drugs, such as weight loss drugs, Viagra, and hair treatment drugs, targeting Arabic-speaking users. The attack scenarios can be modeled as follows:

1. Categorizing attacks by their influence and type of violation (such as causative integrity attacks).
2. Identifying the attack's settings, which include an adversary's goal, knowledge, and capability.
3. Defining the attack strategy, which includes potential attack steps.

3.3.2 Potential Attack Scenarios

Here, attacks against Twitter spam detectors are categorized into four groups: causative integrity, causative availability, exploratory integrity, and exploratory availability attacks. Four attack scenarios are provided, and different examples for each category are presented. Some spam tweets were extracted from tweets posted on Arabic hashtags to show how an adversary can manipulate tweets. Figure 3.1 shows a diagram of the proposed taxonomy of potential attacks scenarios against Twitter.

3.3.2.1 Causative Integrity Attacks

Example 1: Poisoning Attack.
In this attack scenario, an adversary attempts to influence training data to cause new spam to bypass the classifier as false negatives. The settings of the attack scenario are as follows: *The adversary's goal* is to compromise the integrity of Twitter spam detectors, and the attack specificity can be either targeted or indiscriminate. *The adversary's knowledge* is assumed to be perfect (white-box setting). In terms of *the adversary's capability*, it is assumed that the adversary capable of influencing the training data. After defining the attack scenario's setting, the next step is the attack strategy. A potential attack strategy is as follows:

- As the adversary's knowledge of the detection system is considered to be perfect, it is not necessary to send probing tweets to gain knowledge.

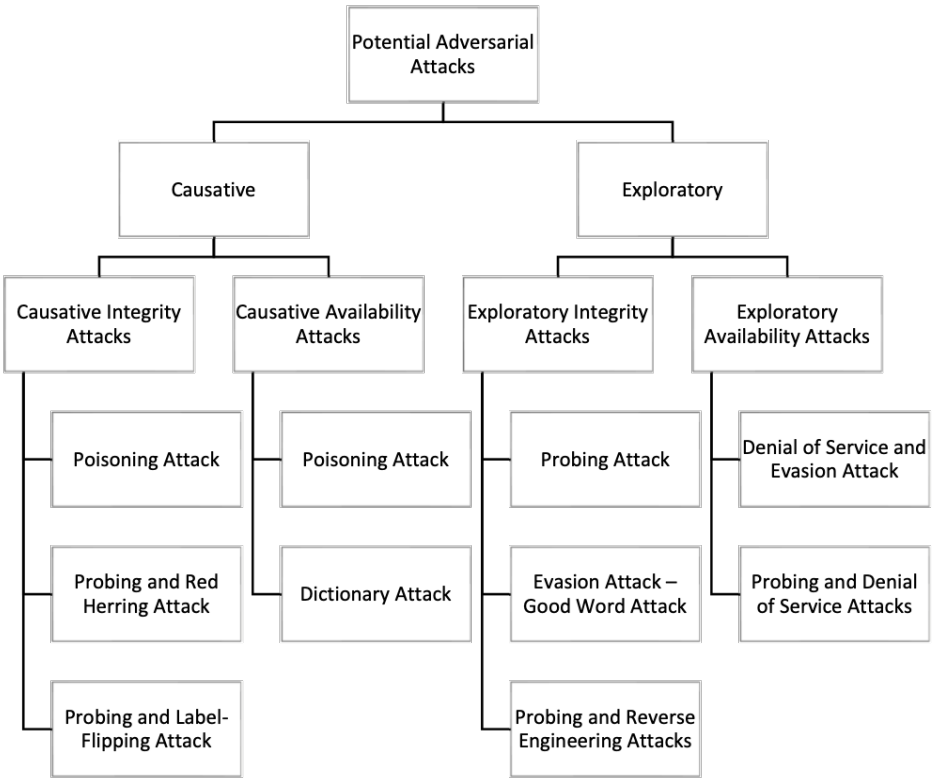


Figure 3.1: Diagram of the proposed taxonomy

- The adversary would carefully craft a large number of malicious tweets.
- The crafted tweets must resemble non-spam tweets and include both spam components, such as malicious URLs, and non-spam components or words (see Figure 3.2).
- The adversary would then post these tweets randomly using different trending hashtags and hope that these malicious tweets are used by Twitter when retraining their system.

Figure 3.2 shows an example of a spam tweet that has been carefully crafted and can be used to poison training data. The spam tweet mimics non-spam tweets by avoiding the inclusion of any spam words, telephone numbers, or hashtags. In addition, the account resembles a legitimate user's account by having a decent number of followers and friends, a profile photo, and a description. This spam tweet bypasses Twitter's spam detector and could be used for retraining the classifier.

Example 2: Probing and Red Herring Attack

As in [205], in this attack scenario, the adversary's aim is to mislead Twitter's spam detectors by influencing training data. *The adversary's goal* is to compromise the integrity and privacy of Twitter's spam detectors, and the attack specificity can be either targeted or indiscriminate. *The adversary's capability* is similar to the previous example. However, *the adversary's knowledge* about Twitter's spam detectors is assumed to be zero (black-box setting). With these scenario settings, a potential attack strategy is as follows:

Figure 3.2: A spam tweet resembling a non-spam tweet to poison training data

- As the adversary has zero knowledge about the detection system, sending probing tweets to gain knowledge is required (privacy violation).
- A probing attack is an exploratory type of attack, and will be discussed in the next section.
- The adversary would craft samples with spurious or fake features and post these samples on trending hashtags to trick Twitter's spam detectors into using these samples for retraining.
- If Twitter spam detectors are trained on these samples, the adversary will discard these spurious features in future tweets to bypass the classifier.

Figure 3.3 (a) number of tweets that have a phone number has increased on Twitter, some proposed spam detectors suggest using a phone number as an indicator of spam tweets. However, detectors suggest using a phone number as an indicator of spam tweets [4, 122]. However, Figure 3.3 (b) shows how the adversary can trick Twitter into using a phone number as a feature and avoid including phone numbers in his spam tweets. Instead, the adversary includes a phone number inside an image to evade detection.

Example 3: Probing and Label-Flipping Attack

The aim of this attack scenario, as in [163], is to cause misclassification by injecting label-flipped samples into training data. The settings of the attack scenario are as follows: *the adversary's goal* is to violate the integrity and privacy of Twitter's spam detectors, and the attack specificity can be either targeted or indiscriminate. *The adversary's capability* is similar to that in the previous example. However, *the adversary's knowledge* is assumed to be zero (black-box setting). According to the scenario's settings, a potential attack strategy is as follows:

- As the adversary has zero-knowledge about the detection system, sending probing tweets to gain knowledge is required (privacy violation).

Figure 3.3: A spam tweet containing a spurious feature (a mobile number). (b) A spam tweet with a mobile number inside an image to evade detection

- A probing tweet (Figure 3.5) helps the adversary to learn how the classifier works; on this basis, the adversary can craft malicious tweets.
- Depending on the knowledge that the adversary gains, he or she can either flip the nearest or farthest samples from the deployed classifier's decision boundary.
- If the adversary did not learn more about the classifier, he or she can randomly flip the label of some tweets.
- He or she then randomly posts these tweets using different trending hashtags and hopes that these malicious tweets are used by Twitter when retraining their system.

3.3.2.2 Causative Availability Attack

Example 1: Poisoning Attack

In this type of attack, an adversary tends to influence training data to either subvert the entire classification process or to make future attacks (such as evasion attacks) easier. The settings of the attack scenario are as follows: *The adversary's goal* is to violate the availability of Twitter, and the attack specificity can be either targeted or indiscriminate. *The adversary's knowledge* is assumed to be perfect (white-box setting). In terms of *the adversary's capability*, it is assumed that the adversary is capable of influencing the training data. After defining the attack scenario's setting, the next step capable of influencing the training data. After defining the attack scenario's setting, the next step is the attack strategy. A potential attack strategy is as follows:

- As the adversary's knowledge about the detection system is considered to be perfect, sending probing tweets to gain knowledge is not required.
- The adversary carefully crafts a large number of misleading tweets that consist of a combination of spam and non-spam components.

- The adversary needs to contaminate a very large proportion of training data for this attack to be successful. Using crowdsourcing sites or spambots to generate contaminated tweets helps the adversary to launch such an attack.
- The last step is to post these tweets randomly using different trending hashtags so that they quickly spread in the hope that Twitter will use them when retraining their system.

Example 2: Dictionary Attack
In this attack, as in [129], an adversary aims to corrupt the classification process by influencing training data and lead future legitimate tweets to be misclassified. The settings of the attack scenario are as follows: *The adversary's goal* is to violate the availability and integrity of Twitter spam detectors, and the attack specificity can be either targeted or indiscriminate. *The adversary's knowledge* is assumed to be perfect (white-box setting). In terms of *the adversary's capability*, it is assumed that the adversary is capable of influencing the training data. After defining the attack scenario's setting, the next step is the attack strategy. A potential attack strategy is as follows:

- As the adversary's knowledge about the detection system is considered to be perfect, sending probing tweets to gain knowledge is not required.
- On the basis of the adversary's knowledge, he or she builds a dictionary of words or phrases that are frequently used by legitimate users and uses this to craft malicious tweets.
- The adversary posts tweets that contain a large set of tokens (e.g., non-spam words, phrases, or tweet structure) from the dictionary in trending hashtags.
- If these tweets are used to train the system, non-spam tweets are more likely to be classified as spam because the system gives a higher spam score to tokens used in the attack.

Figure 3.4 shows how a causative availability attack can affect Twitter spam detectors. The two spam tweets remain undetected for a long period of time because of the attack. As mentioned earlier, availability attacks overwhelm the system, which leads to difficulty in detecting spam tweets. The spam tweet on the left-hand side of the image below contains a very common spam word and should be very easily detected by the classifier, yet as a result of the attack, the tweet remains posted for longer than 52 min. In addition, the spam tweet on the right-hand side remains undetected for longer than 5 h, which is very long.

3.3.2.3 Exploratory Integrity Attack

Example 1: Probing Attack
In this attack scenario, the aim is to learn or expose some of the deployed classifier's functionalities without any direct influence on the training data. The settings of the attack scenario are as follows: *The adversary's goal* is to compromise the privacy of Twitter's spam detectors, and the attack specificity can be either targeted or indiscriminate. *The adversary's knowledge* is assumed to be zero (black-box setting). As in [40], in terms of *the adversary's capability*, it is assumed that the adversary is only capable of influencing the testing data. After defining the attack scenario's setting, the next step is the attack strategy. A potential attack strategy is as follows.

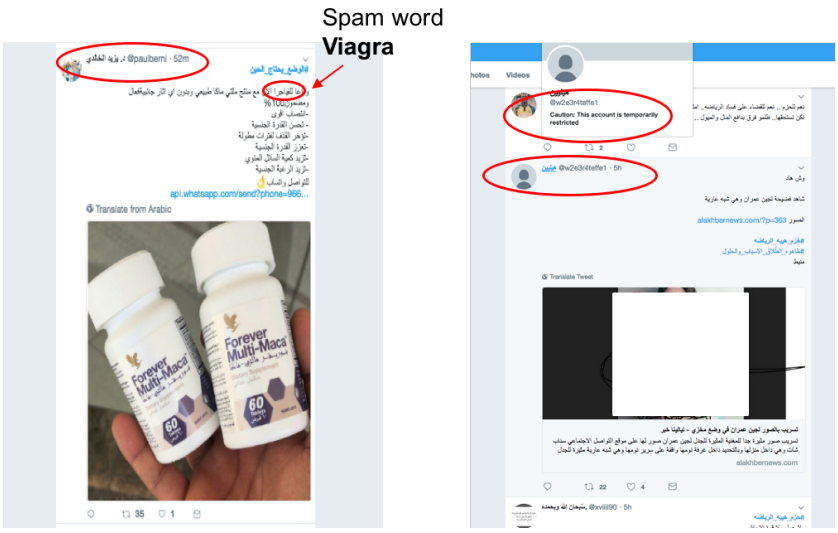


Figure 3.4: Spam tweets bypass the detection system as a result of the availability attack

- As the adversary does not have sufficient knowledge of how the Twitter spam detector works, sending probing tweets to gain knowledge is required.
- The adversary sends a large number of tweets, each with different features, to learn about the system (see Figure 3.5).
- Using the information that is learned, the adversary carefully crafts tweets to evade detection.

Figure 3.5 shows an example of three spam tweets advertising the same weight-loss products. However, the adversary uses different features in each tweet. The first tweet consists of text, a URL, and an image, and the second has text and an image. The last one contains text only. The goal here is to learn how the classifier works. For example, if the first tweet is detected, the adversary will learn that a blacklist of URLs could be one of the features used by the classifier.

Example 2: Evasion Attack – Good Word Attack

In this attack scenario, the aim is to evade being detected by the deployed classifier without any direct influence on the training data. The settings of the attack scenario are as follows: *The adversary's goal* is to compromise the integrity of the Twitter spam detector, and the attack specificity can be either targeted or indiscriminate. *The adversary's knowledge* is assumed to be perfect (white-box setting). In terms of *the adversary's capability*, as in [40], it is assumed that the adversary is only capable of influencing the testing data. After defining the attack scenario's setting, the next step is the attack strategy. A potential attack strategy is as follows:

- As the adversary's knowledge of the detection system is considered to be perfect, sending probing tweets to gain knowledge is not required
- Using his or her knowledge, the adversary carefully crafts tweets by modifying and obfuscating spam words (such as "Viagra") or the tweet's features to evade detection (such as the number of followers) (see Figure 3.6).

Figure 3.5: An example of a probing attack

Figure 3.6 shows a spam tweet that has been carefully crafted to evade detection. The adversary avoids including any spam words in the text. Instead, the tweet contains a description of the drug (Viagra) and the spam word was inserted inside an image.

Example 3: Probing and Reverse Engineering Attacks

Evading the classifier without influencing the training data is the aim of this attack scenario. The scenario's settings are as follows: *The adversary's goal* is to violate the integrity and privacy of Twitter's spam detectors, and *the attack specificity* can be either targeted or indiscriminate. *The adversary's capability* is similar to that in the previous example, but the adversary's knowledge about Twitter's spam detectors is assumed to be zero (black-box setting). From these scenario settings, a potential attack strategy is as follows:

- As the adversary has zero knowledge about the detection system, the first step is to send probing tweets to learn how the system works (privacy violation).
- Using the exploited knowledge, the adversary builds a substitute model that can be used for launching different exploratory attacks [6].
- Once the substitute model is built, the adversary crafts different spam tweets to evade detection, and spam tweets that successfully evade the model are used against the Twitter spam detector.

Figure 3.6: Spam image tweet crafted to evade detection

3.3.2.4 Exploratory Availability Attack

Example 1: Denial of Service and Evasion Attack

In this attack scenario, the main aim is to evade been detected by sending a large number of adversarial spam tweets to overwhelm the deployed classifier without any direct influence on the training data. The settings of the attack scenario are as follows: *The adversary's goal* is to violate the availability and integrity of the Twitter spam detector, and the attack specificity can be either targeted or indiscriminate. *The adversary's knowledge* is assumed to be perfect (white-box setting). In terms of *the adversary's capability*, as in [40], it is assumed that the adversary is only capable of influencing the testing data. After defining the attack scenario's setting, the next step is the attack strategy. A potential the attack strategy is as follows:

- As the adversary has perfect knowledge about the detection system, sending probing tweets to gain knowledge is not required.
- Using the gained knowledge, the adversary carefully crafts spam tweets. As the adversary cannot influence training data, the adversary crafts tweets that require more time for the classifier to process, such as image-based tweets [21].
- The adversary then floods the system (for example, by using a particular trending hashtag) with spam tweets to prevent users from reading non-spam tweets and this causes difficulty in detecting spam tweets.

Figure 3.7 shows an example of an availability attack, in which the adversary uses a different

account to post a large number of spam tweets that only contain an image. As mentioned earlier, image processing overwhelms the deployed classifier and causes a denial of service. In this kind of attack, the adversary may use crowdsourcing sites or spambots to generate spam tweets.

Figure 3.7: An adversary uses a hashtag to flood the system with spam tweets

Example 2: Probing and Denial of Service Attacks

The aim of this attack scenario is similar to that in the previous example, but the scenario's settings are slightly different. *The adversary's goal* is to violate the integrity, availability, and privacy of Twitter's spam detectors, and the attack specificity can be either targeted or indiscriminate. *The adversary's capability* is similar to that in the previous example, but *the adversary's knowledge* about Twitter's spam detectors is assumed to be zero (black-box setting). From these scenario settings, a potential attack strategy as follows:

- he adversary has zero knowledge about the detection system, the first step is to probe the classifier with some tweets to learn how it works.
- Using the exploited knowledge, the adversary crafts a large number of spam tweets and posts them with a specific hashtag to cause denial of service and make future attacks easier [6].

All attack examples can be either targeted (if an adversary focuses on a specific spam tweet, such as URL-based spam or weight-loss ads) or indiscriminate (if an adversary targets multiple types of spam tweets, such as URL-based tweets and advertisements). Although the presented adversarial

spam tweets look very similar to spam tweets that target users, this special type of spam tweet needs to be studied more because it aims to subvert Twitter spam detectors. Table 4.8 summarizes the taxonomy of potential attacks.

Table 3.2: Taxonomy of potential attacks against Twitter spam detectors

Type of Influence	Potential Attack	Security Violation	Specificity
Causative	Poisoning Attack	Integrity	Targeted/ Indiscriminate
	Probing and Red Herring Attack	Integrity & Privacy	
	Probing and Label-Flipping Attack	Integrity & Privacy	
	Poisoning Attack	Availability	
	Dictionary Attack	Availability & Integrity	
Exploratory	Probing Attack	Privacy	Targeted/ Indiscriminate
	Good Word Attack	Integrity	
	Probing and Reverse Engineering Attacks	Integrity & Privacy	
	Denial of Service and Evasion Attack	Availability & Integrity	
	Probing and Denial of Service Attacks	Availability, Integrity & Privacy	

3.3.3 Potential Defense Strategies

This section discusses some possible defense strategies against adversarial attacks that can be considered when designing an adversary-aware spam detector for Twitter. Some of the popular defense methods proposed in the literature are discussed in the context of Twitter spam detection.

3.3.3.0.1 Defenses Against Causative Attacks Existing approaches defending against causative attacks focus on filtering or screening all the training data before using them to update a deployed classifier; such approaches include RONI, data sanitization techniques, and bagging of classifiers. Although these methods have been shown to reduce the influence of contaminated samples on training data, in some cases in which contaminated samples overlap with untainted samples, discriminating between the two becomes very difficult [57]. Some recent studies have suggested using a data collection oracle to retrain a deployed classifier [229, 146]. However, trusting an oracle to label training data could be problematic. The authors in [195] stated that using crowdsourcing sites to label data might produce noisy data, thus increasing complexity. Furthermore, Song et al. [243] added that adversaries can increase the popularity of malicious tweets by using artificial retweets generated by crowdsourcing workers. Thus, developing a fully automated model that can filter these poisoned samples is important. Nowadays, the trend is toward fully automated systems to eliminate human errors. However, the above defense methods require human interventions.

3.3.3.0.2 Defenses Against Exploratory Attacks As mentioned in Section 3.2, the common defense methods against exploratory attacks are disinformation and randomization. The goal in disinformation methods is to hide some of the important information about the system from an

adversary. Although determining the features used by the classifier is not difficult, manipulating or mimicking all of these features may be impossible for an adversary. Some features can be neither manipulated nor mimicked. In [129, 274], the authors found that time-based features (such as account age) are unmodifiable. Furthermore, the authors in [271] discussed how altering some features comes at a cost, while others cannot even be altered. For example, the number of tweets, the number of followers, and the number of following are features that can easily be mimicked, and they might cause the adversary to create a large number of accounts and buy lots of friends. On the other hand, profile and interaction features are much harder to alter. Consequently, considering the robustness of selected features and applying the disinformation method when designing a spam detector could help reduce the effect of adversaries' activities. However, this cannot stop determined adversaries from trying every way possible to accomplish their goals [223]. Furthermore, as stated in [8], relying on obscurity in an adversarial environment is not good security practice, as one should always overestimate rather than underestimate the adversary's capabilities. In randomization, the defender's aim is to mislead the adversary by randomizing the system's feedback. Unlike the disinformation method, this strategy cannot prevent adversaries from exploiting some information about the detection system, but it makes it harder for them to gain any information [24], especially on Twitter, where the adversary uses the same channel as that used by benign users to discover the detection system. This makes randomization methods less effective against exploratory attacks on Twitter.

However, some recent studies have proposed an approach that can detect adversarial examples using the deployed classifier's uncertainty in predicting samples' labels. In [229], the authors used MCS (predict and detect) for detecting adversarial activities. Each classifier detects samples that lie within the classifier's region of uncertainty (blind spots), and the classifier needs to use its best guess. Then, if there is disagreement between the two classifiers' output, the sample will be tested with labeled samples for confirmation.

3.4 Summary

The use of ML models in cybersecurity systems has become very common. As spam on Twitter is considered to be an adversarial problem, investigating the robustness of ML-based spam detectors is very important. Adversaries tend to launch different types of attacks to evade detection by influencing the deployed detector either at the training or test phase. Recent studies have shown an increased interest in studying the robustness of ML models in cybersecurity systems such as IDSs, malware detection, and email spam filters. However, the robustness of Twitter' spam detectors has not been evaluated sufficiently. Since the definition of attack scenarios is ultimately an application-specific issue [39], identifying possible attack scenarios against Twitter spam detectors is needed.

The main contribution of this chapter is the provision of a taxonomy of potential adversarial attacks against Twitter spam detectors and a discussion on possible defense strategies that can reduce the effect of such attacks. Examples of adversarial spam tweets that can be used by an adversary are provided. This study is the first step toward evaluating the robustness of Twitter spam detectors to adversarial tweets, as it identifies potential attacks against them. Examples of possible attacks against Twitter spam detectors are based on common frameworks proposed

in [41, 22, 23] for assessing the performance of ML models in adversarial environments. In addition, defense methods that have been commonly proposed in the literature and ways to deploy these methods in the context of Twitter spam detection are discussed. This chapter answers the first research question (*RQ1*). This chapter shows how adversaries can devise adversarial examples to attack Twitter spam detectors.

Throughout the research in this chapter, a number of challenging issues are mentioned; future research needs to focus on addressing them. Detecting spam images is an ongoing problem, as the processing of images overwhelms the detection system and affects its performance. Adversaries can take advantage of this issue and launch more sophisticated attack. Furthermore, spam detectors designed for spam campaigns may fail to detect single spam attacks and vice versa. This issue can also be exploited by adversaries when attacking spam detectors. Most proposed defense strategies can make attacks against Twitter spam detectors very hard for adversaries, but, as most adversarial attacks are non-intrusive [225], they cannot completely prevent attacks from happening.

Chapter 4

Spam Campaigns Detection: a Robust Feature for Improving the Robustness to Adversarial Hijacked Accounts

We continue investigating the adversarial attacks in Twitter trending hashtags. Practically, we introduce the second contribution of the thesis in this chapter. Our analysis of the targeted spam campaigns in Twitter Arabic hashtags shows that they use a unique type of hijacked accounts as adversarial examples. Consequently, we develop an adversary-aware ML-based detector considering the three key points. We improve the robustness of existing ML-based detectors to the identified *adversarial hijacked accounts* by designing a new feature *avg_posts*. Also, the proposed detector was designed to be adaptable and interpretable to handle *adversarial drift*. (This chapter includes and expands on our unpublished papers:

- Imam, N. and Vassilakis, V., Kolovos D., Towards Designing An Adversary-Aware Detection of Twitter Spam Campaigns *Journal of Digital Forensics, Security and Law*
- Imam, N. and Vassilakis, V., Kolovos D., An Empirical Analysis of Health-Related Campaigns on Twitter Arabic Hashtags *Online Social Networks and Media*

4.1 Introduction

Detecting spam campaigns is challenging, as they are usually designed using complex techniques to impersonate the behaviours of legitimate user accounts [10]. Spam campaigns can create bots that are hard to be distinguished from legitimate users; these bots can easily generate a large number of spam tweets and spread misinformation to create a trending topic. In addition, spam campaigns evolve over time by adopting new techniques to evade detection [74, 78]. Spam campaign designers

use different methods to fool the deployed spam detectors; for instance, using compromised (hijacked) accounts, creating fake accounts, or posting messages with empty content.

Suspicious accounts on social media can be categorised into three groups: human spammers, bots, or cyborgs [70]. *Bots* (also called social media bots or spambots) are accounts controlled by a third party (e.g. software) to generate content and spread messages automatically [84]. These automated accounts have long been used in OSNs, which raises several questions among users [47]. Approximately 15% of all Twitter accounts are bots, which equates to more than 48 million bot accounts, according to a study presented in [262]. It has also been reported that Twitter suspends 9.9 million bots and trolls (i.e. accounts that distort conversations) each week as of May 2018 [10]. While there are many non-malicious social media bots used for news dissemination and legitimate activity coordination, bots are more commonly used for malicious activities [263]. Bots help spam campaigns to achieve scalability through automation. For example, spam campaigns use bots to create fake political support [31], disseminate religious hatred [10], promote terrorist propaganda [104], promote fake financial news [77, 79] and spread healthcare rumours [237]. On the other hand, *Trolls* can be considered as type of cyborgs accounts, which are software-assisted human. Mutlu et al. [201] defines trolls as opinion manipulators accounts. Fornacciari et al. [108] stated that the act of trolling is highly subjective, but the general characteristics of trolls are: disrupting interactions, aggravating interaction partners, and luring them into fruitless argumentation.

Account hijacking has become one of the techniques commonly used by spam campaigns to spread malicious messages. Account hijacking is a valuable technique used by adversaries for several reasons. For example, compromised accounts can exploit trusting social circles, as friends or followers believe that the source is trustworthy [95, 232]. Some incidents, where a malicious party compromises high-profile accounts (e.g., a newspaper, popular brand, or celebrity), highlight the severity of the damage that compromised accounts can cause. One well-known incident is the exploitation of the Associated Press Twitter account, which affected the stock market and the US dollar in April 2013 [232]. Although it has been reported that compromised accounts only exist for a maximum of five days, this short period is often enough for the spammer to achieve their goals [250]. Thus, reducing this time window can lessen the possible consequences. Existing approaches for detecting hijacked accounts, which build a behaviour profile for each user, have some limitations, such as they are not applicable for early detection and are computationally expensive [258].

Adversaries post their malicious messages on twitter hashtags to reach a large number of users in a short period of time using the above different types of spamming activities. Users can use the hashtag symbol (#) to identify a topic and initiate a discussion with others. Once a hashtag starts to be discussed and retweeted by users, it can 'go viral' and become a trend in a given region [190]. Trending hashtags are a primary feature of Twitter, a platform which users regularly visit to get news. Although adversaries can use hijacked accounts to send direct messages, users still can decide whether to open or not by checking the sender's address or the message title. However, the broadcasting nature of hashtags makes them a favourable place for spreading spam since whatever posted under the hashtags will most likely viewed by participants since they have no control over the hashtag. As such, trending hashtags are the logical place to look for spam campaigns, as the posts are persistent and public. The streaming nature of trending hashtags makes the detection of spam is more challenging (e.g,. the fast flow of messages, and changes in the data distribution

over time). Moreover, trending hashtags are adversarial environments, in which adversaries not only try to avoid detection, they may attack the deployed detection system to degrade its performance. Modern ML-based models for detecting spam in trending hashtags face the following challenges: 1) handling the velocity and volume of tweets, 2) being computationally lightweight, 3) being robust against adversarial attacks, and 4) handling the adversarial drift that may occur as a result of adversarial attacks [228, 153].

This chapter presents a study of what appears to be spam campaigns spreading untrustworthy healthcare products using trending Twitter hashtags. The study consists of four phases. First, we analyze a recent hashtag that encouraged users to block health-related campaigns to measure the impact of these campaigns on users. The aim of this part is to answer this research question: (RQ1) *Why is it important to analysis and detect messages posted by Health-related campaigns?*. Second, an empirical analysis of spam messages posted by the targeted campaigns in Arabic trending hashtags was carried out after collecting and building a dataset. Collected messages were manually labelled into two classes: spam and non-spam. This phase aims to answer this research question (RQ2) *why does Twitter spam detection system fail to efficiently detect spam tweets posted by the targeted campaigns?* In the third phase, the characteristics and behaviours of spam tweets posted by the targeted campaigns were analyzed. The results provide some interesting observations on the behaviours of the tweets posted by the campaigns; for instance, a large number of the accounts used in these campaigns were found to be hijacked accounts. Based on this observation, a question naturally arise is: (RQ3) *how can we detect health-related spam tweets posted by hijacked accounts under trending hashtags using features that can be calculated with less computational cost ?* Considering this question help us designing a new feature that can differentiate between legitimate user accounts and hijacked ones in trending hashtags. Finally, an adversary-aware ML-based detector consisting of Multiple Classifiers System (MCS) was developed to detect the targeted spam in trending hashtags considering the robustness to the identified adversarial hijacked accounts, adaptability and interpretability to ensure the model can evolve over time (i.e., handling the adversarial drift). This part of the research aims to answer the following question: (RQ4) *how can we design an adversary-aware detector that can detect spam tweets posted by the targeted campaigns on trending hashtags* The main contributions of this chapter are as follows:

- We analyse behaviors of messages posted by what appears to be spam campaigns in Arabic trending hashtags and we then provide insight into what makes these campaigns difficult to be detected.
- We design a robust feature to improve the detection of the identified adversarial hijacked accounts on Twitter trending hashtags. The new feature was designed based on the following observation: the gap between the creation date and number of total messages posted for legitimate user accounts and the adversarial hijacked accounts are different. This insight was used to design a new feature (*avg_posts*).
- To demonstrate the effectiveness of our designed feature, we employ three feature selection techniques to measure the importance of the new feature. We compare detection results of the identified adversarial hijacked accounts of different supervised and unsupervised ML models

when using the new feature. Also, we evaluate the importance of the new feature using two well known datasets.

- The designed feature *avg_posts*, which is faster to compute compared to features used by users' behavioural-based approaches COMPA [95] and Nautua [202], and which also improves the accuracy with which hijacked accounts can be identified to 73% (up from 19% in [95] and 36% in [202]) without analysing users' tweets history.
- We have shown that being able to detect whether an account is hijacked or not, can improve the accuracy of deciding whether a health-related tweet is spam or not from 86% to 97%.
- We propose an adversary-aware ML-based detector that is robust, adaptable and interpretable. We demonstrate the ability of the proposed detector in handling the adversarial concept drift using the real-world dataset collected from twitter.

The remainder of this chapter is organized as follows. In Section 4.2, we review and summarize the related research on detecting different spamming activities in OSNs and highlight the contribution of our paper. The proposed methodology is described in Section 4.3. Experimental results are presented in Section 4.4, and the analysis of the targeted campaigns is discussed in Section 4.5, followed by conclusions in Section 4.6.

4.2 Related Work

This chapter fits into the broader context of spamming in OSNs, which covers spam bots, trolls and account hijacking. While some related works focus on addressing one of these issues, recent spam campaigns have used different strategies to spread malicious messages, including the use of bots as well as hijacked accounts. Thus, considering relevant studies in the area of spam detection is important when analyzing and characterizing spam activity in OSNs.

4.2.1 Spam Campaigns

The methodologies used for detecting spam campaigns in OSNs involve several steps, one of the most important of which is analyzing user behaviour. Chu et al. [69] proposed a collective detection approach to identify spam campaigns on Twitter. Their methodology involves the following steps. After data collection, tweets were clustered into campaigns and these campaigns were then analyzed for feature engineering. A new feature was designed after finding that spam campaigns send duplicated tweets, unlike regular users. The reason is that it requires effort for spammers to create a large number of fake accounts. Thus, spammers create a number of fake accounts and post duplicated spam messages. This observation led the authors to design a new feature: *Account Diversity Ratio*. A similar observation was found in [288], where Zhang et al. show that spammers use bots to auto-post tweets. Bots send duplicated tweets at the same time or over a short period, such as tweets containing the same URL. Lee and Kim [170] proposed a novel approach for detecting malicious Twitter campaigns at the time of their creation. The approach clusters accounts that seem to belong to the same campaigns by analysing accounts names using agglomerative hierarchical

clustering algorithm and measuring the distance between two names. Therefore, the average time intervals between posting duplicated tweets were calculated to detect spam campaigns. Gupta et al. [122] characterized and analyzed spam campaigns that spread spam tweets containing a phone number, such as product advertisements or pornography dissemination. They defined a campaign as a group of posts containing the same phone number. They proposed a collective classification approach, where nodes in a network are classified based on correlation between known and unknown labels, such as nodes connected by the same phone number or URL. Also, Washha et al. [274] designed a robust new set of features that are difficult for spammers to manipulate. They leverage the time properties of accounts, such as account age and posting behaviour features.

Our proposed methodology builds on the methodology proposed by Chu et al. [69]. The new designed feature was inspired by the approach proposed in [274], where the authors leverage accounts' temporal patterns to detect spam campaigns. These new, unmodifiable features have proved to be efficient in detecting spam activities in OSNs. However, they were not designed for real-time detection, such as in trending hashtags, where the stream of messages is fast-flowing. In addition, most of the related works rely on content analysis (e.g., URLs or phone numbers) or network analysis approaches, which are expensive and time consuming. Furthermore, these approaches were developed to detect spam campaigns, but may fail to detect an individual spam tweet.

4.2.2 Spam Bots

Chu et al. [68] conducted one of the earliest studies measuring the differences between humans, bots, and cyborgs (either a bot-assisted human or a human-assisted bot). The study characterized tweeting behaviour, tweet content, and account properties for the three classes. The classification model consists of an ensemble of classifiers and a Linear Discriminant Analysis (LDA) as a decision-maker, an entropy component for detecting accounts posting times and periodicity, an ML-based component for classifying tweets' textual content, and an account property component that uses URLs, follower-to-friend ratios, and tweet source. Miller et al. [196] proposed an approach for detecting spam tweets in real-time. They handle the streaming nature of tweets by using two stream clustering algorithms: StreamKM++ and DenStream. These algorithms have been trained to cluster normal tweets and to filter anomalies. A web page was built by Davis et al. [84] that allows users to check whether a Twitter account is a bot or not. The web page takes an account screen name as an input to retrieve the account's recent posting history, mentions, and other activities. Then, the classification model computes a bot-likelihood score based on six feature categories: network, user, friend, temporal, content, and sentiment. This web page, later named *Botometer*, has also been used for dataset annotation, bot account verification, and model evaluation in other research works [10, 94, 280]. The authors found that bots use screen names composed of 15 characters randomly generated from alpha-numeric strings (e.g., Ly5PU3QegWlvHjC). They found that Japanese and Arabic bots are the most frequent accounts of randomly generated screen names. The evolutionary nature of spambots forces practitioners to switch from the traditional bots detection techniques that focus on analysing the misbehaviour using supervised approaches to unsupervised ones [73]. Chavoshi et al. [58] proposed the first unsupervised bot detector called *DeBot*. The near real-time system groups correlated users' accounts based on their warping correlations and calculates the cross-user activity. Another study

proposed a detection model that can distinguish bots by their screen name [30]. Cresci et al., [76], propose a social fingerprinting technique that enables distinguishing between spambots and regular user accounts throw analysing the digital DNA and the similarity between the two groups. Also, Mazza et al. [191] proposed an unsupervised-based approach called RTbust, which is a group-based technique for detecting bots. The study leverages unsupervised feature extraction and clustering. The authors used an LSTM autoencoder extract latent feature vectors from the retweet time series of accounts and then clustered them by using a hierarchical density-based algorithm.

While most of the aforementioned studies addressed bots in OSNs English language spaces, very few studies focus on Arabic bots. Some recent works investigate whether bot behaviours are influenced by language and cultural variations [47, 100, 200]. An analysis of Arabic bots' behaviours was conducted in [47]. The authors developed an Arabic bot detector, and the results for the collected dataset showed that Arabic bot behaviours are not dissimilar to that of English bots. A honeypot dataset was also created in [200] for detecting Arabic bots.

Although detecting bots in OSNs has long attracted researchers' attention, only a limited number of studies focuses on detecting bots in trending hashtags. The latter requires designing a tweet-level detection approach, whereas most studies have adopted user-level approaches. Entropy metrics have previously been shown to have a positive correlation with bots, as these metrics indicate that an account displays automation behaviour. However, computing these metrics is time consuming, so they are inapplicable to spam detection in trending hashtags. Also, the method proposed in [30] for detecting bots based on screen names does not take into consideration that spammers can use hijacked accounts to fool this kind of detection models. Similarly, studies that focus on analysing the relationships between users' accounts, such as retweeting [191], correlated users' accounts [58], accounts' digital DNA [76] can be compromised by hijacked accounts. These approaches cluster or group accounts based on the relationships or retweeting activities; adversaries can fool these approaches by hijacking accounts that belong to non-spam clusters or groups. Also, social fingerprinting techniques may fail to detect hijacked accounts as these accounts' digital DNAs are similar to non-spam accounts ones. Although unsupervised approaches proved to out preformed traditional supervised approaches [73], our new designed feature for detecting hijacked accounts can be used by supervised approach and unsupervised approaches as well. Additionally, the identified adversarial attack uses legitimate users' accounts to escape detection unlike most of spambots that try to evade detection by their human-like disguise [103].

4.2.3 Account Hijacking

One of the first models designed for compromised account detection, called COMPA, was proposed by Egele et al. [96]. This model uses a supervised approach to build behaviour profiles (e.g., posting, retweeting) for accounts and then detects anomalies. M. Nauta [202] used tweets language, posting time, number of tweets per day and other features for detecting hacked accounts. Another study used users' writing style as a feature for recognizing compromised accounts [131]. Similarly, Seyler et al. [232] developed a framework for detecting compromised accounts by examining the textual patterns in the account's history. If an account's textual output, such as language, changed, the account would be considered compromised. In addition, A study by Viswanth et al. [265] proposed

an unsupervised model for detecting anomalous users on Facebook. The authors employed Principal Component Analysis (PCA) to build behaviour models for normal users. VanDam et al. [260] proposed an unsupervised framework (CADET) that does not rely only on the textual features of accounts. Instead, it uses a combination of textual and meta data features for detecting compromised accounts. They rely on sources commonly used by compromised accounts, time of the day compromised accounts posts at, and compromised accounts' frequent locations.

Most existing works in detecting hijacked accounts have focused on using supervised and unsupervised approaches. However, some recent works have used Deep Neural Networks (DNN) approaches. Karimi et al. [258] proposed a DNN-based model for detecting compromised accounts, developing an End-2-End Compromised Account Detector (E2ECAD) that consists of three components: temporal features, user context, and network feature extractors. E2ECAD uses Long Short-Term Memory (LSTM) networks to capture accounts' temporal features, such as post time and post source. User context uses feature embedding to capture the semantics of words and phrases, and the doc2vec approach for converting users' posts into a fixed vector representation. Finally, the authors constructed social graphs to extract accounts' network features. Similar to previous works, E2ECAD uses all of users' posts to learn users' behaviours. In addition, a deep learning framework was developed by VanDam et al. [258] for detecting compromised posts by utilizing the user and post features simultaneously. This framework consists of tweet content-based and user-based encoders to learn features embedding, and a neural network that calculates the residual errors for the encoders. Since user features may not be sufficient for making predictions, a small set of the user's initial tweets is used for better predictions.

Compared with previous studies, our research provides a more comprehensive approach for studying spam campaigns in OSNs. Methodologies, features, and classification algorithms used for detecting spam bots, trolls, and hijacked accounts were taken into account to better understand spam campaigns in OSNs. The approach proposed in [258] [149] are the most close approaches to ours, but we focus on detecting spam tweets that are posted under trending hashtags. The proposed approach exploits features of hijacked accounts that are hard for adversaries to manipulate. We considered spam tweets posted by hijacked accounts as an adversarial examples, which none of the related studies have discussed. Treating hijacked accounts as an adversarial examples requires designing a detection model that is robust against adversarial attacks, adaptable to possible evolving attacks, and human-interpretable. Previous studies have detected hijacked accounts at the user level, which is not an applicable approach for detecting spam in a fast-streaming data environment (e.g., trending hashtags). Building a behaviour profile for each users as in [95, 202, 131, 259], or detecting whether there have been any changes (e.g., in writing style and language [96, 232]) is computationally expensive and time consuming approach. Using time of day or tweet's source as features [260] is not a robust approach, as adversaries can easily manipulate these kinds of *weak features*. Regular users change their behaviours over time, so using these weak features makes the deployed detection model sensitive to concept drift [258, 138]. Models that build behavioural profiles or network/social graphs are able to identify hijacked accounts after a change in behaviour of regular accounts is detected. However, as hijacked accounts have a short lifetime, early detection is crucial, something that these approaches may not be able to achieve.

4.2.4 Robust Features

One of the techniques that is usually used to counter spammers' evasion tactics (adversarial attacks) is designing new and robust features. Yang et al [278] defined a robust feature as a type of features that is either difficult or expensive for the adversaries to evade. The authors designed new features for distinguishing spam and legitimate accounts. They considered an account's tweet number as it reflects the account level of activity. Also, they designed three new neighbor-based features: average neighbors' followers, average neighbors' tweets, and followings to median neighbors' followers. Chen et al [62] used account age and other 11 features to detect real-time spam on Twitter. Sicilia et al, [237] proposed different new features including avg number of followers, followings, status, and registration age. Mateen et al [190] proposed a technique that uses a combination of user-based, content-based and graph-based. For example, FF ratio, hashtag ratio, mentions ratio, and reputation. Madisetty and Desarkar [185] develop an ensemble model that uses deep learning and feature-based methods for detecting spam at tweet level. Authors use accounts age, status counts, and other features. Albadi et al [10] redesigned new content-based features, such as avg emoji, numerics, links, and punctuations along with tweet-based and account-based features for detecting hateful messages in social media. In this study, we designed a robust feature for detecting hijacked accounts by leveraging accounts' temporal patterns. Unlike related studies, we use account age and status (i,e number of posts) to design a new feature.

4.2.5 Spam Detection in Adversarial Setting

Limited studies have investigated the detection of spam in an adversarial setting. Wang et al. [271], evaluate the vulnerability of an ML detector, which is designed to detect spam on Weibo (Chinese Twitter) generated by malicious crowdsourcing workers using evasion and poisoning attacks. Their focus was on adversaries that use crowdsourcing sites to launch evasion and causative attacks. The authors conclude that the fewer knowledge adversaries have about the system, the harder it is for them to evade detection. Another study by Nilizadeh et al. [206] analysed the robustness of a Twitter spam detector called *POISED* against evasion and poisoning attacks. POISED is designed to distinguish between spam and non-spam messages based on the way messages spread on Twitter instead of looking at accounts' or messages' characteristics. A novel proactive approach for detecting spambot was proposed by Cresci et al. [80]. The authors use accounts digital DNA and genetic algorithms simulations to anticipate adversarial behaviours of spambots. Also, Cresci et al. [81] propose GenBot, a novel genetic algorithm designed to generate social spambot that simulate the behavior of human accounts.

The analysis of an ongoing health-related campaigns in trending Arabic hashtags, revel a new adversarial attacks. Unlike previous studies of spam detection in an adversarial setting that synthetically generate adversarial examples and proposed a detection methods, we found a new adversarial examples that use hijacked accounts to fool spam detectors. Specifically, the identified adversarial hijacked accounts can fool ML-based spam detectors that analysis tweet statistical features and hijacked accounts detectors that utilize user's behavioural-based approach. These ML-based spam detectors trained to classify messages through analysing accounts features, such as number of friends/ followers, or account age and content features (e.g. number of words, or number of hashtags etc.).

These detectors would classify hijacked accounts as non-spam based on their statistical features. Also, the hijacked accounts used by the targeted campaigns are either inactive that do not contain a large number of tweets or accounts with only spam tweets. User's behavioural-based detectors can not detect these kind of hijacked accounts efficiently as their accounts do not have enough variations.

Our approach for detecting spam tweets posted by health-related campaigns in Twitter's hashtags builds on a large body of prior work which focuses on characterizing spam campaigns and bots. Previous approaches, [237, 274, 149, 258, 185], where the authors leveraged temporal patterns in users' accounts to detect spam campaigns, are the closest works to the present study. One limitation of these studies is that they focused their analysis on spam campaigns, without taking into consideration hijacked accounts, which are widely used by spam campaigns. Several researchers have studied the characteristics of hijacked accounts and proposed different approaches for detecting these accounts with fairly high accuracy [232, 250, 258, 96, 202, 259]. However, these approaches may not capable of detecting hijacked accounts efficiently in the context of trending hashtags, since they focus on analyzing account posting history, which delays detection. The analysis in this research is based on a collection of tweets extracted from different Arabic trending hashtags, covering sports, politics, entertainment, and other topics, over the period of a year (May 2018 – November 2020). Three essential points were considered when designing our adversary-aware ML-based detector, which makes the approach different to related works: its robustness, adaptability and interpretability. (1) It is robust to the identified adversarial hijacked accounts, as it utilizes a new designed feature, which is hard to be manipulated by adversaries. (2) It consists of MCS to ensure the adaptability to adversarial concept drift, which can occur as a result of adversarial attacks. (3) It uses a FRB classifier to make the detector human-interpretable, so security analysts can debug it when it is needed.

4.3 Our Method

This section describes the methodology used for developing the adversary-aware ML-based detector proposed in this chapter. It follows methods commonly used in the literature, but, in addition, the possible presence of adversaries was considered in each step. The methodology consists of three main steps: campaigns analysis, robust feature extraction, and adversary-aware detector development.

4.3.1 Analysis of Healthcare Ads Campaigns

Here, we focus on analyzing healthcare ads campaigns, which can help to design a diverse detection system, as these campaigns tend to use different spamming methods. Spam campaign analysis can provide a better understanding of possible spamming tactics, which can help in designing a more efficient detection system. A natural question arises: *Why is it important to analyse and detect messages posted by Health-related campaigns?* These campaigns violated Twitter rules[1] by sending bulk of advertisements and advertise a third party products. Also, the Saudi food and drugs authority has issued a warning statements about some of the products advertised by these campaigns (see Figure 4.1). Also, we found that these campaigns use hijacked accounts to spread their spam

[1] https://help.twitter.com/en/rules-and-policies/twitter-rules

messages. Additionally, to answer the above question, we measure impact of these campaigns on users' experience by analyzing a recent trending hashtag. Details and results discussion are presented in the following subsection.

Figure 4.1: A statement of the Saudi food and drugs authority

4.3.1.1 Impact of Health-Related Spam Campaigns

The negative effects of spreading misinformation, such as the propagation of misleading healthcare claims, manipulation of the stock market, religious hatred on OSNs, have been proven in the literature [10]. In July 2019, Facebook published a statement saying that the company would be adding two new News Feed algorithms to reduce sensational or misleading health claims [130]. The statement also mentioned that these health-related claims were against a 'free for all' principle as these posts prey on vulnerable users. Hence, the impact of health-related ads campaigns on Twitter users' experience was considered in this chapter. We analyzed a trending hashtag (#حمله ـ تبليك ـ منتجات ـ التخسيس) that encouraged users to block weight loss ads campaigns (#blocking_campaign_of_slimming_products). Twitter sentiment analysis, which is a process of categorizing users' opinions toward a particular topic, was carried out. Sentiment analysis utilizes Natural Language Processing (NLP) for extracting subjective information (e.g., features) from text or learning polarity of words [216, 156]. We used two methods for analyzing the hashtag: manual labelling and TextBlob [2].

The results of the hashtag sentiment analysis are presented in Table 4.1, which shows that the hashtag contained 415 tweets: 145 positive, 30 negative, and 240 neutral tweets. Tweets that supported the hashtag were labelled as positive tweets. A large number of positive tweets consisted of users providing different methods for how to block these health-related campaigns. One of the tweets that provided instructions on how to block such campaigns was retweeted 50 times. Several

[2]https://github.com/adhaamehab/textblob-ar

Table 4.1: Hashtag sentiment analysis.

	Manual Labelling	TextBlob
Total Tweets	415	415
Positive Tweets	153	145
Negative Tweets	22	30
Neutral Tweets	240	240

supportive tweets were found in the hashtag; for example, some users stated that they had been waiting for such a hashtag to share their opinions about these health-related campaigns. Negative tweets are messages that either disagreed with the hashtag or expressed the belief that blocking does not work. For instance, a number of tweets were complaining about not being able to block these campaigns and demanding action from Twitter. Others mentioned that blocking is not enough to stop these campaigns and advised users to report the campaigns to Twitter. Neutral tweets are a type of message that contain only images/videos, unrelated tweets, or spam messages posted by health-related campaigns. One of the neutral tweets claimed that the campaigns had stolen their accounts, while another tweet stated that the campaigns hijack inactive accounts. One of the accounts alleged that these campaigns are used intentionally to distract users. As the hashtag trended, 45 health-related ads tweets were found in the hashtag. Given that the hashtag was created to block health-related campaigns, and became a trending hashtag, this demonstrate the extent of the effect these spam campaigns have on users' experience.

The results of this analysis, which answers *RQ1*, indicate that these campaigns have some of the *Trolls'* behaviours. One of the main characteristics of using trolls is to distort the information flow [194], which has been reported by users who participate in this hashtag. These campaigns distort users from reading others users' posts. Also, prevent users from participating in some hashtags by posting healthcare products ads that contain pornography images.

4.3.1.2 Data Collection

Most of the datasets used in related works [237, 274, 30] are not publicly available. Publicly available datasets [138, 62] contain fewer attributes, and crawling further attributes (e.g., status and number of hashtags) is required to capture the complete set of features needed for this study. Moreover, most of the corresponding accounts in publicly available datasets are suspended and thus there is no information available to retrieve. Hence, a new dataset was built for the purpose of the current study. The process of developing the dataset followed the same steps discussed in previous studies. The dataset was collected by using Twitter's standard search Application Programming Interference (API)[3], which provides access to 1% of Twitter's global streaming data, for the period from May 2018 to November 2020. The tweets were collected from trending hashtags in Saudi Arabia in different domains, including social, political, and entertainment, to avoid any possible bias. The datasets were collected in different time-windows during a two-year period. We extract tweets from different

[3]https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

Arabic hashtags, but the chosen ones were not taken equally from the hashtags. The reason for not choosing tweets equally from the hashtags are that there are a lot of duplicated tweets and we were hunting the adversarial hijacked accounts, which can not be found in all hashtags.

Table 4.2: Examples of spam Tweets posted the health-related campaigns

Original Tweet	English Translation	Label
حصريامنتج القدرة الجنسيةآمن صحي للطلب والاستفسار عبر الواتساب في أول تعليق	Exclusively Sexual Ability Product Healthy Safe To request and inquire via WhatsApp in the first comment	spam
لحل جميع المشاكل الجنسية للرجال منتج طبيعي ليس له أعراض جانبية	To solve all men's sexual problems a natural product that has no side effects	spam
منتج بروتينات لبناء عضلات الجسم ليصبح جسمك رياضي ورشيق اطلب المنج الآن عبر رسائل الخاص أو عبر ال	Product RG Proteins to build the muscles of the body To make your body sporty and agile Ask for your product now via private messages or via	spam

4.3.1.3 Ground Truth

Various approaches for labelling collected datasets have been proposed in the literature, such as creating honeypots, checking whether accounts have been suspended by Twitter, or manual labelling. Since the targeted health-related spam campaigns use different spamming tactics (e.g., bots and account hijacking), the manual labelling method was chosen to establish ground truth. Some previous works have created artificial datasets to simulate compromised accounts on Twitter [232, 256]. However, creating artificial accounts was not considered for this study, as identifying health-related spam is a relatively easy task since they are a common occurrence in Arabic trending hashtags. The dataset was labelled into two classes: spam and non-spam. Tweets advertising health-related content were classified as spam. We excluded other types of spam tweets, such as religious propaganda, pornography, or finance, since they are not the target of our work. After we manually label the dataset, we recruited two annotators that are Arabic native speakers. One of the annotators is a PhD student specialist in pragmatics and she has experience in Twitter data annotation. The second annotator is a Computer Science student who has worked on annotating Twitter datasets. The tweets in the dataset were shuffled and our labels were removed. Also, we remove accounts screen names, user names and IDs, so the annotators were provided tweets' textual content only. The annotators were received brief guidelines and given three days to finish the annotation. In addition, it was checked whether tweets labeled as spam are suspended by Twitter. Table 4.2 presents some examples of our dataset's tweets. The guidelines provided to the annotators are as follows:

1. *spam*: tweets that advertise healthcare products, such as Diet, weight loss, skin, sex, hair, etc.

2. *non-spam*: any tweets that do not advertise healthcare products.

Figure 4.2: An account has incompatible username and screen name

Figure 4.3: Three accounts having the same picture

To prove the annotations step reliability and validity, the inter annotator agreement (IAA) was used. In addition to measuring the annotations reliability, IAA shows how clear the annotation guidelines are and whether the annotation task is reproducible. In this study, we use Fleiss's Kappa [107] that is a statistical IAA measure for assessing reliability between more than two annotators. We calculated Fleiss's Kappa for the 2,509 tweets that were annotated by three annotators (i.e., two recruited annotators and the authors of this paper). The Kappa was 0.96 which is interpreted as almost perfect agreement according to Landis and Koch [166].

4.3.1.4 Campaign Analysis

Campaign analysis is an important step as it helps in understanding spam account behaviours and with the feature engineering task. After annotating the dataset, we found that there are some spam tweets posted by legitimate accounts violate Twitter rules. Also, we found some spam tweets posted by accounts that have incompatible username and screen name (see Figure 4.2), and others were posted by accounts that have similar account's picture (see Figure 4.3). Most importantly, these campaigns exhibit different posting behaviours, unlike the spam campaigns discussed in the literature, which post at a certain time of a day. In particular, we found that, unlike other generic spam campaigns, these campaigns post under trending hashtags and, in some cases, cause hashtags to trend. Similarly, a previous study discussed similar spamming behaviours found in Arabic trending hashtags [47]. These spam campaigns use different tactics to avoid detection. In our previous studies [142, 139], we have found that these campaigns post tweets with different content (e.g., textual, media, or a mix) to fool the deployed classifiers. One of the strategies that is used by these campaigns, besides creating fake accounts and bots, is to use hijacked accounts. Several characteristics were identified that can reveal whether an account has been hijacked or not. For example, a change in language or a change in posting behaviour. Notably, these campaigns were found to use old accounts that have very few tweets in their timelines. Figure 4.4 presents an example of a health-related spam tweet posted by a hijacked account that was created in 2016 and has only 20 tweets.

Figure 4.4: A hijacked account created in 2016 with a few total number of tweets

Figure 4.5: An example of a hijacked account posted in two diffident languages.

Identified hijacked accounts are slightly different to typical account hijacking scenarios. Figure 4.5 shows a tweet from a hijacked account, where the last tweet posted by the account owner was in 2013; then, in 2019, a health-related tweet was posted by the account. The reason for this gap could be either because these accounts have been suspended by Twitter, or they have been abandoned by their owners. Figure 4.6 shows a typical account hijacking scenario. As can be seen, legitimate accounts initially post a series of tweets before being hijacked. Then, the attacker will post some tweets, as shown by the grey area. Once the account owner discovers that his or her account was hijacked, they report this to Twitter or inform their friends/followers that their account has been hijacked. However, a different hijacking scenario is seen in hijacked accounts found in Arabic trending hashtags. It seems that some of these accounts tend to have not been used for a long time period before being hijacked. The time interval between the last tweet posted by the account owner and the first tweet posted by the attacker after taking over the account is significant.

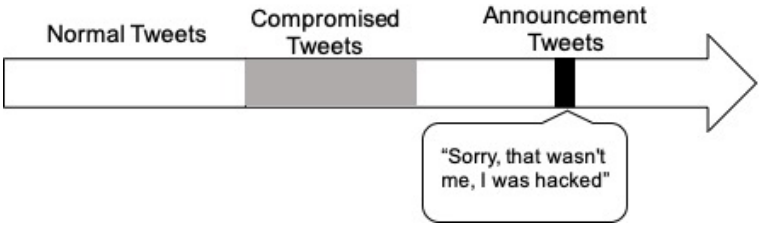


Figure 4.6: A typical account hijacking scenario.

Hijacked accounts used by health-related campaigns were found to be old accounts with only a few tweets, unlike legitimate accounts, where the number of tweets increases as the accounts get older. To prove this hypothesis and design a new feature, we used the Spearman's rank test [244], a well-known statistical test, to measure the strength of the correlation between *account age* and *number of posts (status)*. The results of the Spearman's test showed a correlation coefficient of 0.4619019 and a p-value of 2.2e-16, and a correlation coefficient of -0.03352242 and p-value of 0.625

for non-spam and spam tweets, respectively. In other words, the test demonstrated that there is a positive correlation between the two features for non-spam accounts. As the account age increases, the number of posts (status) also increases, whereas a negative correlation was found in spam accounts. Figure 4.7 presents a scatter graph plotting the correlation between x-axis (account_age) and y-axis (status) for non-spam tweets. As can be seen in the graph, an upward slope in the selected trend-line, which is *the Coefficient of Determination (R^2)*, is an indication of a positive correlation.

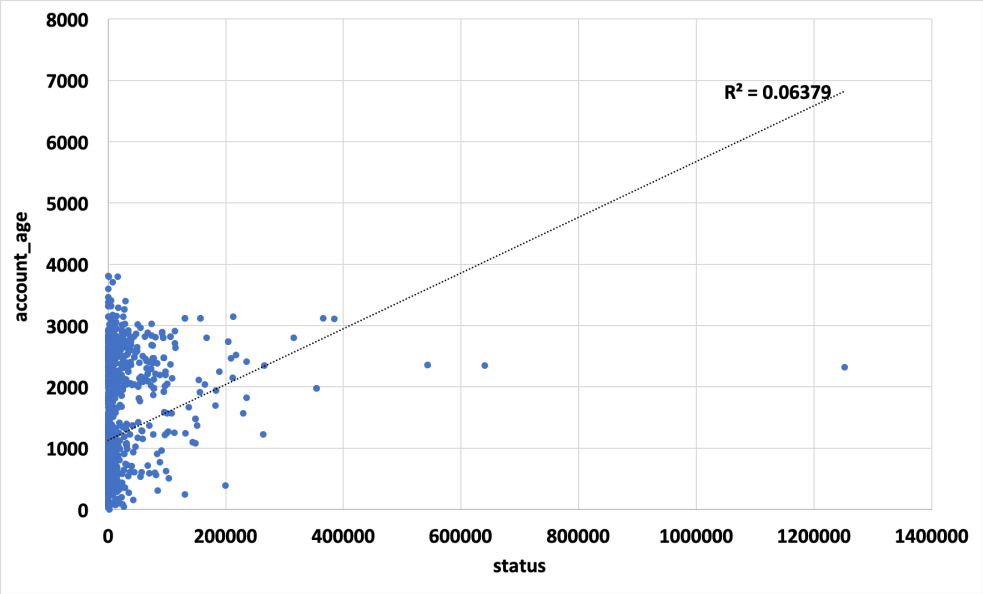


Figure 4.7: Correlation between *status* and *account_age* for non-spam tweets.

Another important task was to explore the collected data to see if the extracted features could be redesigned leveraging the above observation to detect the *adversarial hijacked accounts* at tweet-level. To calculate the average number of posts for an account's timeline, two features were used: 'status' (total number of posts) and 'created at' (account creation date). First, account age was computed by calculating the difference between accounts' creation date and annotation date, formally expressed as: $account_{age} = Date_{annotation} - Date_{creation}$. Second, the average number of posts for each Twitter account in the collected dataset was calculated by dividing the number of posts by account age. This is expressed formally as $avg\_posts = \frac{status}{account\_age}$. The following section will provide more insight into feature engineering and the impact of each feature on the final prediction. Note, our feature was designed to detect messages at a tweet-level, which makes it applicable for the streaming nature of trending hashtags. We do not consider the daily rate of posting as it requires analysing accounts' posting history. Instead, we calculate the average number of post for each tweets posted under a hashtag.

4.3.2 Feature Extraction

This section describes the features extracted from Twitter for use in the data analysis task and to build the proposed detector. Most of these features have been used in the literature [260, 178], but

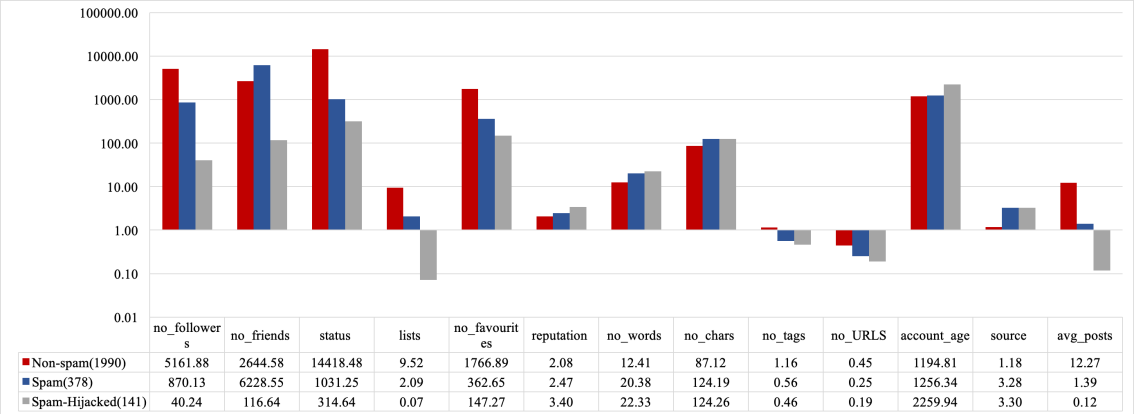
a new designed feature was also added. Features were categorized into three groups: account-based, tweet-based, and content-based. Table 4.3 presents the extracted features with short descriptions.

Table 4.3: Features extracted from Twitter for data analysis and feature selection tasks.

Feature Category	Feature Name	Description
Account-based	account_age	The number of days since the creation of an account
	no_followers	The number of followers of an account
	no_friends	The number of friends an account has
	no_favorites	The number of favorites an account has received
	no_lists	The number of lists an account is a member of
	reputation	The ratio of the number of followers to the sum of followers and friends
	status	The number of tweets an account has
	protected?	Whether the account is private or public
	verified?	Whether the account has been verified by Twitter
	geo?	Whether the account has the geotagging feature enabled
	time_zone	What time zone is used by the account
	language	What language the account uses
	user_name	Name chosen when creating the account
	screen_name	User screen name
	description	user bio (maximum 160 characters)
	avg_posts	The average number of tweets an account has since creation date
	is_suspended?	Whether the account is suspended by Twitter
	is_bot?	Whether the account uses a randomly generated 15 character string for the screen name
	is_hijacked?	Whether the account is a hijacked account
Tweet-based	source	Type of device used for tweeting
	created_at	Tweet creation date
	is_retweet?	Whether the post is a retweet
Content-based	tweet	Tweet content
	emoji	a standardized set of pictographs
	no_words	The number of words in a tweet
	no_chars	The number of characters in a tweet
	no_hashs	The number of hashtags in a tweet
	no_urls	The number of URLs in a tweet

Additionally, we have designed four new features, which are also presented in Table 4.3, to help understanding the characteristics of the targeted campaigns. These four features were used only for the data analysis purpose. The first feature *is_bot?* was used to measure the number of bot accounts in these campaigns. The definition of bots follows the observation in [30], which considers accounts that use 15 randomly generated characters in their screen names as bots. Secondly, *is_suspended?* feature was used to quantify the number of accounts suspended by Twitter. Also, to check whether a type of a tweet (e.g. original tweet or retweet) can help with the classification task, *is_retweet?* feature was used [261, 149]. Most importantly, *is_hijacked?* feature used to count the number of hijacked accounts. Accounts that have been inactive for a long period of time and suddenly start posting health-related ads, or old accounts that have very few health-related ads are considered as

hijacked accounts.



	no_followers	no_friends	status	lists	no_favourites	reputation	no_words	no_chars	no_tags	no_URLS	account_age	source	avg_posts
Non-spam(1990)	5161.88	2644.58	14418.48	9.52	1766.89	2.08	12.41	87.12	1.16	0.45	1194.81	1.18	12.27
Spam(378)	870.13	6228.55	1031.25	2.09	362.65	2.47	20.38	124.19	0.56	0.25	1256.34	3.28	1.39
Spam-Hijacked(141)	40.24	116.64	314.64	0.07	147.27	3.40	22.33	124.26	0.46	0.19	2259.94	3.30	0.12

Figure 4.8: A general statistical distribution of the dataset

Figure 4.8 shows the general statistical distribution of the collected dataset. The latter consists of 2,509 tweets that are grouped in two classes: 1,990 non-spam tweets and 519 health-related spam tweets. The spam tweets includes 141 tweets that are posted by hijacked accounts. The figure shows a comparison between the mean of non-spam, spam, and spam tweets posted by hijacked accounts in the 13 features. The new feature *avg_posts* shows a significant difference between the mean of non-spam tweets and spam tweets posted by hijacked accounts as well as some other features.

4.3.3 The Development of Adversary-aware Detector

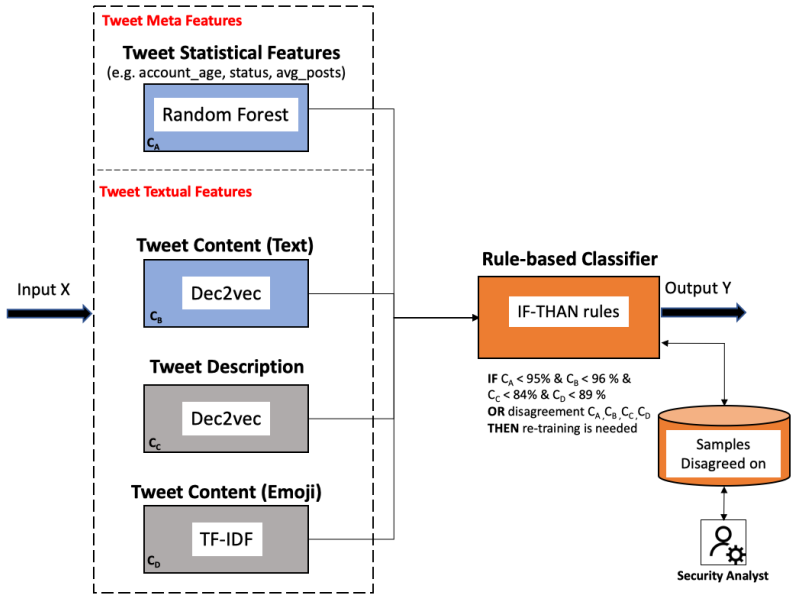


Figure 4.9: The overall structure of the proposed adversary-aware detector

Recent studies show that when ML-based models are used in cybersecurity systems, they become

vulnerable to various forms of adversarial attacks [43, 7, 36]. A taxonomy of possible adversarial attacks against spam detectors was proposed in our previous study [142]. Thus, the proposed detector was designed for an adversarial environment, in which adversarial drift may occur because of adversaries' constant attempts to compromise the deployed system. The design of our spam detector is inspired by some of the models discussed in the related work section that utilize Multiple classifier systems (MCS) are also known as ensemble classifies [228, 68, 260, 149, 220]. Related studies focus on detecting hijacked accounts and tweets using MCS to capture different features (content-based or meta-based features) and to improve the detection accuracy. Table 4.4 presents different detection models proposed in the literature for detecting hijacked accounts and tweets using MCS. Unlike existing approaches, our adversary-aware ML-based detector consists of four classifiers: one meta feature-based and three textual feature-based classifiers. The proposed detector considers four modalities of data: tweets' statistical features (e.g., account_age, status, avg_posts, etc.), tweets' textual content, tweets' description, and tweets' emojis content. Diversity is an important characteristic of MCS as measuring the diversity helps prunes the classifiers [186]. According to [161], the best method to measure the diversity of an MCS type of models is measuring the disagreement. Thus, the output of the four classifiers are feed into a Fuzzy Rule Based (FRB) classifier for measuring the disagreement and making the final prediction. FRB classifiers consist of a set of IF...THEN rules that are transparent and interpretable by humans. FRB classifiers are widely used to deal with uncertainties or to process non-stationary streaming data [90, 14, 120]. Although the design of traditional FRB classifiers requires a number of handcrafting functions, assumptions and patterns to be selected, our adversary-aware ML-based detector utilizes an FRB classifier for integrating the outputs of the MCS in a way that does not require a large number of rules. The main reasons for using FRB are to detect possible adversarial drift that may occur as a result of evolving adversarial attacks (adaptability) and to make the detector interpretable. Related studies use majority vote or softmax function for the final prediction of the ensemble classifiers' outputs, which may can not detect adversarial drift and default to be debugged by a security analyst. We believe that when designing a spam detector for adversarial sittings, it is crucial to consider how the detector will operate under a new adversarial attack. Thus, considering the robustness to an identified adversarial attack, the adaptability to evolving attacks and interpretability to a security analyst are important. The process of our detector involve three steps. First, the input x is classified by the four classifiers; the output of these classifiers is either 0 (non-spam) or 1 (spam). The output of these classifiers is then examined by the FRB classifier, and the final decision of this classifier is the output y. An overview of the proposed adversary-aware detector is presented in Figure 4.9. The following subsections will provide a more detailed description of the detector and its components.

4.3.3.1 Meta Feature-based Classifier

This classifier makes its predictions based on input statistical features, such as number of friends or followers; this classifier uses a total of 13 numerical features. The statistical features of tweets can help distinguish spam from non-spam. However, in the real world, these features may change in unpredictable ways over time, and several vectors may cause data distribution to drift over time. For example, users may change their behaviours over years, or adversaries may purchase followers

Table 4.4: Related studies use multiple classifiers for detecting compromised accounts

Related Work	Multiple Classifier Models
COMPA: Detecting Compromised Accounts on Social Networks [95]	Two classifiers: Mandatory consists of time of the day, source, proximity and language, Optional includes links, direct interaction, and topic.
End-to-End Compromised Account Detection [149]	End-to-end framework E2ECAD consists of three components – Temporal Feature Extraction, User Context, and Network Feature Extraction. The results of the components are concatenated and the softmax function is utilized for the final prediction.
CADET: A Multi-View Learning Framework for Compromised Account Detection on Twitter [259]	Unsupervised framework consists of four auto-encoders. The framework encoded each feature of users' accounts independently using four views (i.e. encoders). Finally, a multi-view decoder is used to aggregate the reconstruction errors of the encoders.
You have been CAUTE! Early Detection of Compromised Accounts on Social Media [258]	Compromised Account User Tweet Encoder CAUTE consists of two encoders: tweet2user and user2tweet. The encoders transform lexical and meta features of a tweet into more informative and less noisy features. Res2class a fully-connected neural network layer uses the residual errors of the encoders to predict if the post is compromised or not.

to impersonate legitimate users. Thus, depending solely on statistical features is not an efficient approach for detecting spam. Instead, we have used features that are hard to modify, and added a textual feature-based classifier to support the detector's final decision. Section 4.4 provides more details about the features used in this classifier

4.3.3.2 Textual Feature-based Classifiers

Three text classifiers were used for analysing tweet's textual content and emoji and account's description. Since one of the goals of the targeted campaigns is to speared untrustworthy healthcare products, they mimic legitimate companies' or organizations' accounts by adding descriptions to their accounts. The results of our analysis of spam campaigns' descriptions shows that 9 out of 519 tweets have an empty description. Also, these accounts use some terms frequently (see Figures 4.10 and 4.11).

Figure 4.10: Word-cloud of most frequent spam words in tweets' content

Figure 4.11: Word-cloud of most frequent spam words in tweets' description

Although the raw features (e.g., the text messages and meta-features) of users' tweets are often

sparse and noisy [258], some of this noise is purposely added to affect the detection accuracy of the deployed classifier. For example, the targeted spam campaigns tweets tend to include emojis and misspellings in the tweets' content and tweets' accounts descriptions. Some recent studies have investigated the visual aspects of the tweets, including emojis, which are ideograms used to visually complement the meaning of tweets' content [89, 155, 128]. We incorporate emoji knowledge to improve the overall detection accuracy of the proposed adversary-aware detector and to add one more defensive layer against possible adversarial attacks. The analysis of the targeted campaigns shows that they use different emojis than legitimate users. For example, we found that approximately 70% of spam tweets include emojis, while 30% of non-spam tweets use emojis. Also, the analysis shows that the top 15 emojis belonging to spam and non-spam are different (see Figure 4.12). For capturing the textual features of tweets, we use three classifiers: Doc2vec for tweets' textual content, Doc2vec for tweets' description and TF-idf for tweets' emojis. To extract the textual features, NLTK Python Library [46] was used. The text classification models were chosen through experiments, which are discussed in Section 4.4.

Figure 4.12: Top 15 Emojis used in spam and non-spam tweets.

4.3.3.3 Fuzzy Rule-based Classifier

Here, a set of rules that depend on the outputs of the four classifiers are defined. The main reasons for using this classifier is to make sure that the detector can evolve over time in the face of emerging attacks. Specifically, the classifier was designed considering the adaptability and interpretability to handle possible adversarial drift that may occur as a result of adversarial activities [228]. The FRB is a preferable solution for such cases, as it provides interpretability which enables interactive debugging [220, 216]. Most importantly, the FRB helps improving the process of decision-making as it take a form of rules that interpretable by humans (e.g. security analyst) [120]. We categorized the four classifiers into two groups as proposed in [95]: *Mandatory* and *Optional*. The outputs of classifiers C_A and C_B are mandatory since a regular tweet is most likely to have statistical features (e.g. number of followers) and content (e.g. text, hashtags, or URLs). On the other hand, the classifiers C_C and C_D are optional as we have found in our dataset that some tweets have an empty description and others do not use emojis. The output of the mandatory classifier is given a higher weight when making the final prediction of an input. Also, the optional classifier C_C is given a higher weight than the classifier C_D because our analysis shows that few tweets do not have descriptions. To this end, human-in-the-loop (HITL) is included in this classifier to debug the detector if an adversarial drift occurs. In adversarial drift, adversaries cause the drift intentionally by manipulating the malicious class only to degrade the deployed classifier and to avoid detection by the traditional concept drift detectors [228]. On the other hand, the traditional concept drift in OSNs may occur as a result of an evolution in people's preferences, population changes (e.g. during special events), the complexity of the environment, or any hidden context) [110]. Although, handling

the traditional concept drift is commonly done by retraining the models, it is not the case when handling adversarial drift since a malicious agent is attempting to evade the deployed model [228]. Only when the adversarial drift is detected or the disagreement between the detector's classifiers increases, the security analyst will check the samples that the classifiers disagreed on and debug the classifiers either by retraining or feature engineering if the drift is confirmed. The set of fuzzy rules and experiments results are discussed in Section 4.4.

4.4 Experimental Results

This section presents and discusses our experimental results and evaluation. We run our experiments on Linux Ubuntu 18.04 LTS operating system with Intel(R) Core(TM) i7-8750H CPU 2.20GHz x 12 of 983.4 GB memory. We utilize keras 2.3.1 [4] for implementation.

This section consists of three subsections. Subsection 4.4.1 presents the characterization of the collected data. The results of three feature ranking and selection methods are presented in Subsection 4.4.2. In Subsection 4.4.3, we present different experiments for developing the proposed adversary-aware detector. We evaluate the importance of the new designed feature in detecting the identified adversarial hijacked accounts using supervised and unsupervised learning models. Then, the comparison of our results with SOTA. Finally, we demonstrate how our adversary-aware detector can handle adversarial drift in Subsection.

4.4.1 Data Characterization

Exploring the collected and annotated dataset is an important task for data characterization. In Figure 4.13, we show the number of tweets posted by bots and hijacked accounts in each class, along with the suspension status of these tweets. Additionally, to gain a better understanding of spam campaigns, the four new features discussed in Subsection 4.3.2 were employed to characterize and analyze the collected dataset. First, the dataset was grouped into four categories: tweets posted by suspended accounts, bots, hijacked accounts, and retweet. Then, the data in these groups was analyzed and characterized based on each tweet's temporal features: *status*, *account_age*, and *avg_posts*. Although other features can also help in understanding the data characteristics, the temporal features were chosen because they are difficult to be manipulated by spammers.

4.4.1.1 Suspended Accounts

Here, the new feature *is_suspended* was used to understand a specific characteristic of the collected dataset. Specifically, our aim was to confirm whether tweets labelled as spam are suspended by Twitter and to analyze tweets labelled as spam that had not been suspended. As we identify some bot accounts in the non-spam tweets, we checked whether these bot accounts had been suspended by Twitter. Figure 4.13 shows that more than half of the non-spam bots were suspended (92 out of 149), while 399 out of 519 spam accounts were suspended. Hence, the dataset was divided into: suspended/non-suspended spam and suspended/non-suspended non-spam tweets. Then, we compare the mean of the three features in each group and analyze the results. Table 4.5 presents comparison

[4] https://pypi.org/project/Keras/

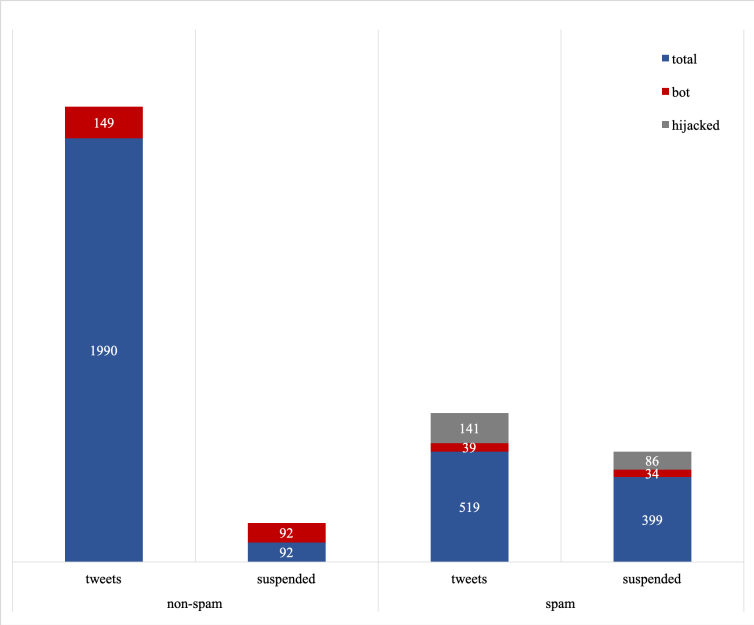


Figure 4.13: Statistical distribution of bots and hijacked accounts

results between suspended and non-suspended accounts for three features: *status*, *account_age*, and *avg_posts*. The presented results are rounded to two decimal places. Specifically, Table 4.5 consists of five columns and four groups (suspended and non-suspended spam and non-spam). The columns present our statistical analysis results for the three features in terms of count (number of tweets), and mean. These results indicate that suspended health-related spam bots have a lower average number of posts (a tweet per day), while the suspended non-spam bots send more than 12 tweets per day. Surprisingly, non-suspended spam accounts were found to have older account age with fewer posts, but non-suspended non-spam accounts had a younger account age with a large number of posts. These are clear indicators of spamming behaviours, as was also explained in Section 4.2.

Table 4.5: A comparison between suspended and non-suspended accounts for the two classes

		status	account_age	avg_posts
non-suspended (spam)	count	120	120	120
	mean	495.87	1136.04	1.05
suspended (spam)	count	399	399	399
	mean	939.02	1647.17	1.04
non-suspended (non-spam)	count	57	57	57
	mean	3931.48	474.94	29.03
suspended (non-spam)	count	92	92	92
	mean	4043.22	375.14	12.09

Comparing the results in each group, Table 4.5 shows that there is no significant difference between the behaviours of suspended spam accounts and non-suspended accounts in terms of the average post per day. The mean of the average number of posts for suspended and non-suspended

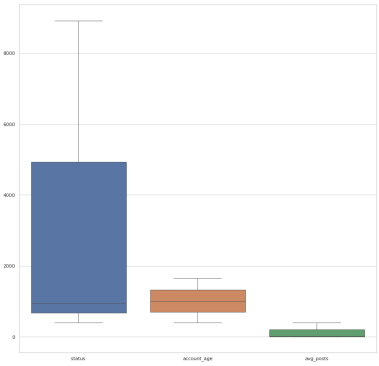


Figure 4.14: Suspended spam

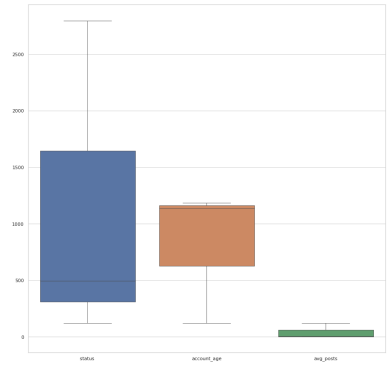


Figure 4.15: Non-suspended spam.

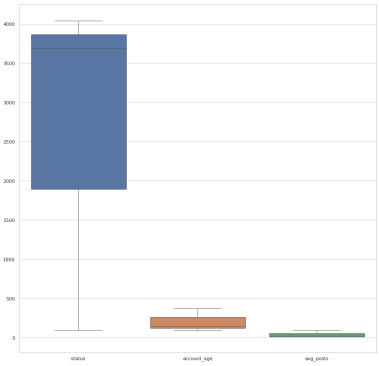


Figure 4.16: Suspended non-spam

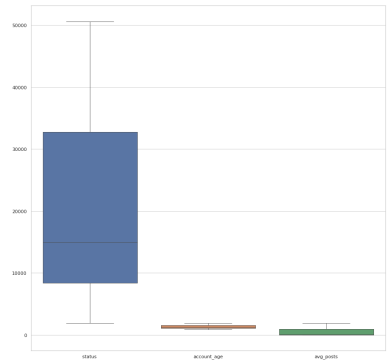


Figure 4.17: Non-suspended non-spam.

spam accounts is approximately one post per day. This suggests that these non-suspended spam accounts should be suspended by Twitter as they have a similar average number of posts to the suspended spam accounts. For non-spam bot accounts, we calculated the mean for the three features and the results show that non-suspended bots post 29 tweets per day whereas the suspended bots post 12 tweets per day. In addition, the suspended bot accounts are younger accounts with a higher number of posts. This means that, even though non-suspended bots are older accounts and have posted fewer tweets, they remain undetected. A possible reason for this could be that the non-suspended bots had been suspended temporarily and then reactivated. Figures 4.14, 4.15, 4.16, 4.17 plot and compare the distributions of Table 4.5 using a graphical display method *boxplots*. The blue, orange, and green boxes identify the three features status, account_age, and avg_posts respectively.

This analysis indicates that a better detection system is needed, as approximately 30% of the spam and bot accounts were found to have not been suspended although the analysis showed that these accounts had exhibited spamming behaviours. Also, 39% of hijacked accounts were still active, whereas 87% of spam bots were suspended. These results indicate that hijacked accounts have different characteristics than bots, which can be detected accurately by bot detectors.

4.4.1.2 Bot Accounts

Different characteristics could be used to identify bots. In this study, we labelled accounts with a user name composed of 15 randomly generated characters as bots (only for analysis purpose) [30].

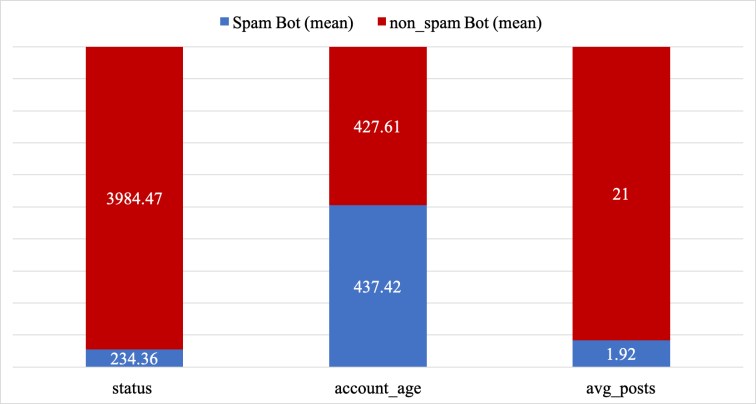


Figure 4.18: Spam bots vs. non-spam bots.

The collected dataset contains 188 bot: 149 non-spam, and 39 spam accounts. Only two of the spam bot accounts were hijacked accounts. After manually checking whether these bot accounts were suspended or not, we found that $\sim$ 62% of non-spam bots had been suspended, while $\sim$ 58% of spam bots were suspended. A comparison between the mean of the three features for spam and non-spam bot accounts is shown in Figure 4.18, where the presented results are rounded to two decimal places. We observe that spam bot accounts have lower average status and average number of posts, while the account ages are almost the same. The mean of the feature *status* for the spam bots is 234, compared to 3,984 for non-spam bots. The mean of *avg_posts* is 1.9 and 21 for spam and non-spam bots, respectively. However, the mean of *account_age* is 437 for spam bots and 427 for non-spam bots. A possible reason for such a significant difference in *status* is that bots are suspension-prone, so these spam bots may post fewer tweets to avoid detection. The difference in *avg_posts* is large because it is affected by the lower number of *status*.

4.4.1.3 Posting Behaviour

Different methods are proposed in the literature to measure an account's level of activity to discriminate between spam and non-spam accounts. For example, analyzing changes in an account's posting language or analyzing the type of posts (i.e., new tweets or retweets). In this study, the type of posts was considered to analyze posting behaviour. We found that spammers tended to post a new health-related and then retweet the post using either bots or hijacked accounts. This behaviour was also discussed in a related study [47]. Also, Wang et al. [270] analysis of malicious messages in OSNs shows that users' interaction (e.g., comment or retweet) are very good metrics for such purposes. Retweeting can speed up the spread of spam messages, so we use a new feature *is_retweet* to analyze tweets labelled as spam. The goal was to see if there is a difference in the behaviour of accounts post retweeted or newly tweeted spam messages. Figure 4.19 presents a comparison between the three features for new tweets and retweeted spam. We observe that 395 out of 519 spam tweets were retweeted, which accounts for 76% of the spam tweet; the remaining tweets were new tweets. The results in Figure 4.19 show that there is no significant difference in *account_age* and *avg_posts* between the retweeted and the newly tweeted types of spam: there is $\sim$ 35% difference between

account ages and $\sim$ 7% for average number of posts, while there is $\sim$ 57% difference in *status* for the two groups. The main difference, in *status*, is understandable as we have found that spammers use some accounts just for retweeting, mentioning, and liking.

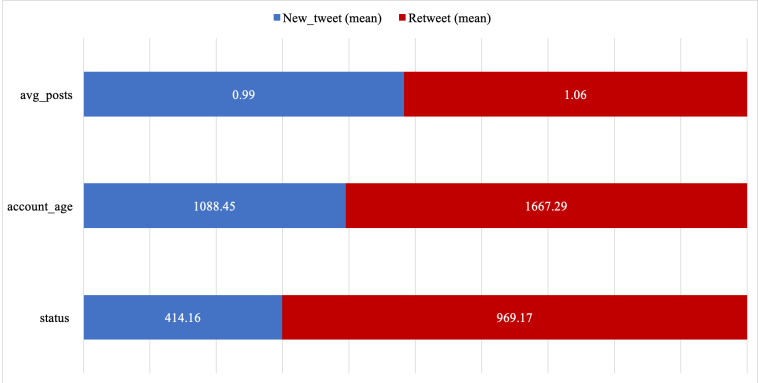


Figure 4.19: Posting behavior.

4.4.1.4 Hijacked Accounts

After investigating health-related spam in the collected dataset, 141 tweets were found to be posted by hijacked accounts. More than half of the hijacked accounts (86) had been suspended by Twitter and only two accounts were bot accounts. These numbers support the observation in [96] that spreading spam by using hijacked accounts is more advantageous for spammers because users tend to trust messages posted by legitimate accounts. The difference between the spam tweets from suspended and non-suspended accounts has been investigated in Subsection 4.4.1.1; in this subsection we analyze the suspension of hijacked spam accounts. A comparison between suspended and non-

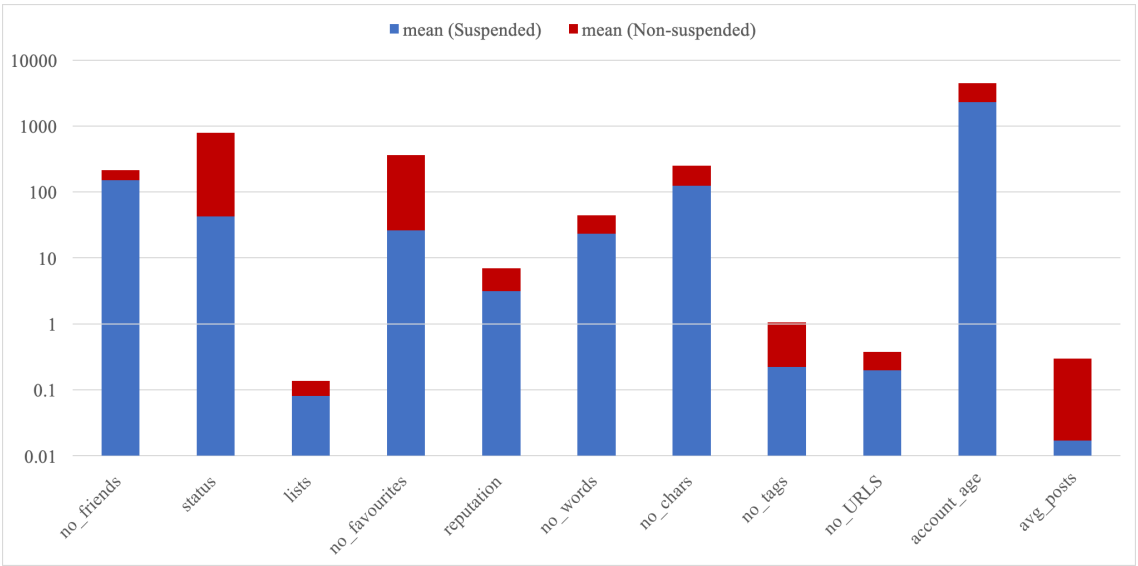


Figure 4.20: A Comparison between suspended and non-suspended hijacked accounts.

suspended hijacked accounts is provided in Figure 4.20. The x-axis lists ten features and the y-axis shows the number of tweets for each feature. The results show a considerable difference between suspended and non-suspended hijacked accounts in both *avg_posts* and *status*, and a small difference in *account_age*. The mean of *avg_posts* for suspended hijacked accounts is ~ 0.02, compared to ~ 0.3 for non-suspended accounts. The mean of *status* for suspended accounts is ~ 43 compared to ~ 740 for non-suspended hijacked accounts. These two features indicate that non-suspended hijacked accounts post more tweets per day, which makes them more similar to legitimate users' accounts. The mean of *account_age* for the two types are also similar. This analysis shows that suspended Twitter accounts have low *avg_posts* and *status*. This means that the longer these accounts remain undetected, the harder it is to distinguish them from legitimate accounts. Additionally, we have found that most of the suspended hijacked accounts are those who have compatible username and screen name as these accounts look as legitimate users' accounts.

4.4.2 Feature Selection and Ranking

Three feature selection and ranking techniques were employed to measure the importance of the new feature (*avg_posts*) to the deployed classifier. Also, this section compares the new designed feature with features that are commonly used in the literature [260, 178]. We use RF for this part of evaluation since it was used widely in the literature [62, 274, 275, 206].

4.4.2.1 Evaluation Metrics

The following evaluation metrics have been used to measure the performance of the classifiers used in the developed adversary-aware detector: accuracy, recall, precision, and F1 score. These metrics, along with their descriptions, are defined in Table 4.6 [212, 175].

Table 4.6: Evaluation metrics

Metric	Description	Function
Accuracy	The ability of a classifier to correctly find spam/ non-spam	$\frac{TP+TN}{TP+FP+TN+FN}$
Recall	The ability of a classifier to correctly find spam	$\frac{TP}{TP+FN}$
Precision	The ability of a classifier to not misclassify spam	$\frac{TP}{TP+Fp}$
F1 Score	The harmonic mean of precision and recall	$\frac{2TP}{2TP+fp+FN}$

Where:
True Positive (TP): is the number of correctly predicted spam, in which the ground truth class is spam and the predicted class is also spam.
True Negative (TN): is the number of correctly predicted non-spam; the ground truth and the predicted values are both non-spam.
False Positive (FP): is the number of incorrectly predicted spam.
False Negative (FN): is the number of incorrectly predicted non-spam.

4.4.2.2 Feature Performance Comparison

First, a method used in [69] for evaluating the discrimination weight for each feature was followed. We used WEKA [124] to calculate the prediction accuracy of each feature via a 10-fold cross-validation test with RF as a classifier. This test provides RF prediction accuracy when using one feature at time. The performance results for the 13 features are presented in Table 4.7. The evaluation metrics used were accuracy, and the Weighted Avg. of True Positive Rate (TPR) and False Positive Rate (FPR). The results show that *avg_posts* achieved the second-best accuracy of 88 and the lowest FPR at 16.9.

Table 4.7: Comparing features in terms of accuracy, TPR, and FPR.

#	Feature	Accuracy %	TPR%	FPR%
1	source	90	90.5	34.2
2	avg_post	88	88.2	**16.9**
3	account_age	87	87.1	26.0
4	no_tags	86	86.5	32.4
5	no_favorites	84	84.1	35.3
6	no_chars	83	83.9	36.8
7	no_friends	83	83.1	42.3
8	no_followers	82	82.4	38.5
9	no_words	81	81.4	44.0
10	statuses	81	81.8	35.4
11	lists	79	79.2	79.3
12	reputation	79	79.2	76.9
13	no_urls	79	79.3	79.3

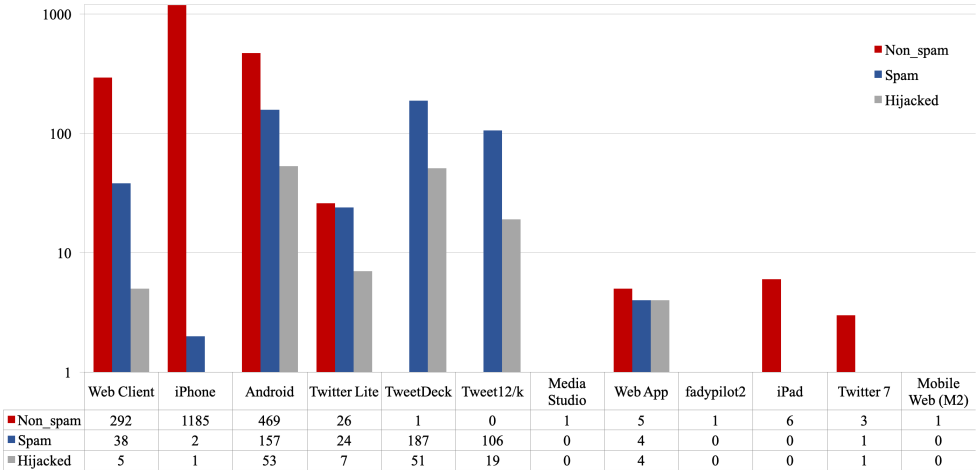
4.4.2.3 Attribute Selection

Second, the attribute selection option provided in WEKA was used to rank the features [124]. The results ranked source as the most important feature and the designed feature fourth among the other 11 features (see Table 4.8).

This is expected, given the fact that less than half of spam tweets were found to be from hijacked accounts. The sources commonly used by non-spam tweets included iPhone, web Clint and Android, whereas TweetDeck, Android and Tweet12/k were used for spam tweets. A comparison between the sources used by the two classes is provided in Figure 4.21. Some previous studies [95, 258, 202, 260, 149, 259, 256] have shown that the post source can help in detecting hijacked accounts. Although *avg_posts* was designed based on the characteristics of hijacked accounts, this feature was found to be informative for distinguishing spam tweets in general.

4.4.2.4 Mean Decrease Accuracy

Finally, to measure the importance of the *avg_posts* feature in determining the accuracy of the model for distinguishing hijacked accounts, the Mean Decrease Accuracy (MDA) was used. MDA is widely used in the literature as a feature selection method to directly measure the impact of each feature



	Web Client	iPhone	Android	Twitter Lite	TweetDeck	Tweet12/k	Media Studio	Web App	fadypilot2	iPad	Twitter 7	Mobile Web (M2)
Non_spam	292	1185	469	26	1	0	1	5	1	6	3	1
Spam	38	2	157	24	187	106	0	4	0	0	1	0
Hijacked	5	1	53	7	51	19	0	4	0	0	1	0

Figure 4.21: Sources used by the two classes.

Table 4.8: RF attribute ranking.

#	attribute	average rank	average merit
1	source	1	0.113
2	account_age	2.5	0.076
3	no_tags	3	0.072
4	avg_posts	3.5	0.069
5	no_chars	5.1	0.052
6	no_favourits	6.1	0.039
7	no_friends	7	0.035
8	no_followers	8.3	0.026
9	no_words	9.2	0.02
10	status	9.3	0.019
11	no_URLS	11.5	-0
12	lists	11.9	-0
13	reputation	12.6	-0.003

on RF accuracy [10, 181]. Code for computing the MDA was adopted from Datadive [5]. The results in Figure 4.22 show that *avg_posts* has the greatest impact on RF accuracy, with a 0.24 importance score. The figure presents the 13 selected features with their respective importance scores. Removing *avg_posts* and *source* decreases the RF's ability to correctly detect hijacked spam (recall) from $\sim$ 94% to $\sim$ 89%.

4.4.3 Adversary-aware Spam Detector

The experiments conducted in this section focus on initially choosing the best ML algorithms, and then training and testing the selected algorithms. As the proposed adversary-aware detector consists of four classification models, separate experiments were performed to choose an ML algorithm for

[5]http://blog.datadive.net/selecting-good-features-part-iii-random-forests/

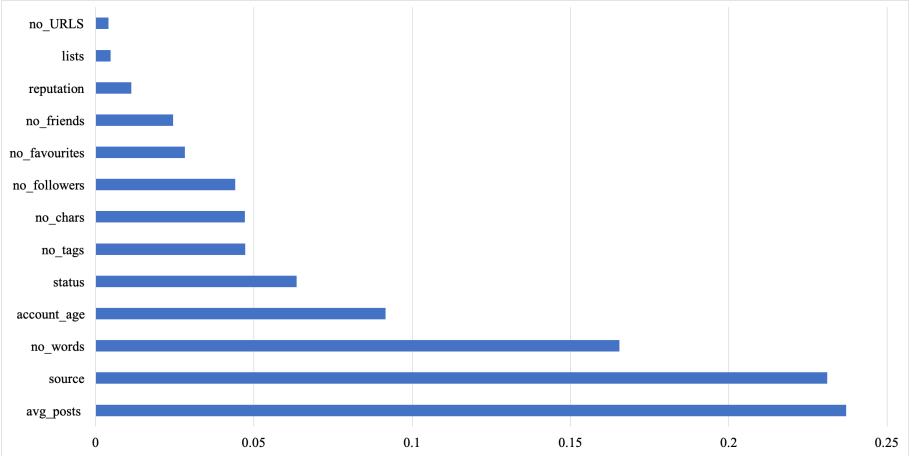


Figure 4.22: The ranking features importance for detecting hijacked accounts.

each model.

4.4.3.1 Meta Feature-based Classifier

The meta feature-based classifier C_A focuses on detecting spam based on tweets' statistical features. After splitting the dataset into 70% training and 30% testing, a comparison of the prediction accuracy of different ML algorithms showed that RF produced the best results. Table 4.9 presents the prediction accuracy, recall, and precision for four ML algorithms. The results show that RF and Gradient Boosting (GB) achieved a similar prediction accuracy of 0.98, whereas accuracy rates of 0.91 and 0.83 were achieved by Support Vector Machine (SVM) and Stochastic gradient descent (SGD), respectively. Finally, RF was chosen for use in this study, as previous research has proved that it is the best choice for a meta feature-based classifier [69, 237, 30].

Table 4.9: A Comparison between different ML algorithms

Classifier	Accuracy	Recall	Precision
RF	0.98	0.95	0.96
SVM	0.91	0.62	1.0
SGD	0.83	0.56	0.54

Additionally, this section focuses on evaluating the effectiveness of the *avg_posts* feature in improving the robustness of meta feature-based classifiers to the adversarial hijacked accounts with two goals. First, we seek to examine how well does the *avg_posts* feature improve the performance of different supervised and unsupervised classifiers in detecting the identified adversarial hijacked accounts. Second, we seek to compare our meta feature-based classifier with SOTA spam classifiers.

4.4.3.1.1 Experimental setup We use three datasets: the dataset collected from Twitter, Gilani-2017 [114] and cresci-rtbust-2019 [191] datasets. The datasets were divided into training and testing datasets. Two versions of the training datasets were used; a training datasets included the

adversarial examples and training datasets that do not include neither bots nor adversarial examples. Hijacked accounts were considered as adversarial examples and used to evaluate the robustness of different ML-based models using different ML algorithms. Description of the datasets used for experiments is presented in Table 4.10. The first dataset was used for evaluating the ML-based detectors when trained by clean dataset. The second dataset, which does not contain adversarial examples was used for evaluating ML-based models when trained in adversarial training fashion. The third dataset was used for testing the ML-based models. Also, the benchmark datasets (Gilani-2017 and cresci-rtbust-2019) were used for evaluating the effectiveness of *avg_posts* by using different datasets.

Table 4.10: Datasets for training and testing experiments

#	Dataset Description	non-spam	spam
1	Training dataset does not include adversarial examples	1796	344
2	Training dataset includes adversarial examples	1990	519
3	Test dataset includes adversarial examples	101	81
4	gilani-2017	1362	1049
5	cresci-rtbust-2019	317	300

4.4.3.1.2 Supervised Models In this set of experiments, we compare the performance of several supervised classifiers, that used in related studies [176, 190, 275], in detecting the adversarial hijacked accounts with and without the new designed feature and the results of the top six were presented. The goal of using different ML algorithms is to evaluate the importance of the designed feature. The ML algorithms used for experiments in this section are as follows:

- Random Forest (RF)
- Gradient Boosting (GB)
- Multilayer Perception (MLP)
- Multinomial Naive Bayes (MNB)
- Stochastic gradient descent (SGD)
- Decision Tree (DT)

First, the ML algorithms were trained by using the first dataset that does not include the adversarial examples and bots (i.e., cleaned dataset). Then, the algorithms were evaluated using the third dataset. Table 4.11 shows that the overall prediction accuracy for most of the ML algorithms increases by at least 2% except for two ML algorithms NMB and DT when using the *avg_posts* feature. Although the detection accuracy of NMB remains the same with and without *avg_posts* and the detection accuracy decreases for DT when using *avg_posts*, these two ML algorithms achieves lower detection accuracy then RF, GB, and SGD. In detail, the recall of RF and MLP decrease by over 10% when removing *avg_posts*, whereas it decreases by 2% for GB. Based on the these results,

we conclude that when the algorithms trained with dataset includes *avg_posts*, their performance on detecting the adversarial hijacked accounts increases.

Table 4.11: A comparison of six ML algorithms trained by the cleaned dataset. Bold indicates significant improvement (degradation)

		precision	recall	F1 score	accuracy
RF	non_spam (with avg_posts)	0.81	0.99	0.89	**86**
	spam (with avg_posts)	0.98	**0.70**	**0.82**	
	non_spam (without avg_posts)	0.75	0.99	0.85	**81**
	spam (without avg_posts)	0.98	**0.59**	**0.74**	
GB	non_spam (with avg_posts)	0.88	1	0.94	**92**
	spam (with avg_posts)	**1**	**0.83**	**0.91**	
	non_spam (without avg_posts)	0.87	0.97	0.92	**90**
	spam (without avg_posts)	**0.96**	**0.81**	**0.88**	
MLP	non_spam (with avg_posts)	0.6	0.87	0.71	**60**
	spam (with avg_posts)	**0.63**	**0.27**	**0.38**	
	non_spam (without avg_posts)	0.57	0.90	0.70	**57**
	spam (without avg_posts)	**0.55**	**0.15**	**0.23**	
NMB	non_spam (with avg_posts)	0.98	0.43	0.59	68
	spam (with avg_posts)	0.58	0.99	0.73	
	non_spam (without avg_posts)	0.98	0.43	0.59	68
	spam (without avg_posts)	0.58	0.99	0.73	
SGD	non_spam (with avg_posts)	0.97	0.92	0.94	**94**
	spam (with avg_posts)	**0.91**	0.96	0.93	
	non_spam (without avg_posts)	0.98	0.9	0.94	**93**
	spam (without avg_posts)	**0.89**	0.98	0.93	
DT	non_spam (with avg_posts)	0.77	0.98	0.86	83
	spam (with avg_posts)	**0.96**	**0.64**	**0.77**	
	non_spam (without avg_posts)	0.81	0.99	0.89	86
	spam (without avg_posts)	**0.98**	**0.70**	**0.82**	

Additionally, we conduct a preliminary experiment on *adversarial training*, by feeding the ML algorithms the second training dataset that includes the adversarial examples and bots. The goal is to evaluate the performance of the ML algorithms that trained on the adversarial training fashion, with and without the *avg_posts* feature. One of the drawbacks of the adversarial training is the generalization error on clean data (i.e., non-adversarial) [257, 279]. Each algorithm was trained with two datasets: one includes the proposed feature *avg_posts* and the second does not include it. Table 4.12 shows that the detection accuracy remains the same for most of the ML algorithms when using *avg_posts* and removing it. However, the recall of MLP and SGD improve by 44% and 14% respectively. Also, DT detection accuracy decreases by 2% when using avg_posts. These results show that the overall detection accuracy of the ML algorithms is not affected when using *avg_posts*, yet it improves the detection accuracy in some algorithms.

Moreover, we investigate the importance of the *avg_posts* feature in detecting the identified

Table 4.12: A comparison of six ML algorithms that trained with dataset include adversarial examples. Bold indicates significant improvement (degradation)

		precision	recall	F1 score	accuracy
RF	non_spam (with avg_posts)	0.97	0.97	0.97	97
	spam (with avg_posts)	0.96	0.96	0.96	
	non_spam (without avg_posts)	0.96	0.99	0.98	97
	spam (without avg_posts)	0.99	0.95	0.97	
GB	non_spam (with avg_posts)	0.93	0.96	0.95	94
	spam (with avg_posts)	0.95	0.91	0.93	
	non_spam (without avg_posts)	0.93	0.96	0.95	94
	spam (without avg_posts)	0.95	0.91	0.93	
MLP	non_spam (with avg_posts)	0.93	0.87	0.9	**89**
	spam (with avg_posts)	0.85	**0.91**	0.88	
	non_spam (without avg_posts)	0.67	0.86	0.75	**69**
	spam (without avg_posts)	0.73	**0.47**	0.57	
NMB	non_spam (with avg_posts)	0.98	0.43	0.59	68
	spam (with avg_posts)	0.58	0.99	0.73	
	non_spam (without avg_posts)	0.98	0.43	0.59	68
	spam (without avg_posts)	0.58	0.99	0.73	
SGD	non_spam (with avg_posts)	0.99	0.85	0.91	**91**
	spam (with avg_posts)	0.84	**0.99**	0.91	
	non_spam (without avg_posts)	0.88	0.86	0.87	**86**
	spam (without avg_posts)	0.83	**0.85**	0.84	
DT	non_spam (with avg_posts)	0.77	0.98	0.86	83
	spam (with avg_posts)	0.96	**0.64**	0.77	
	non_spam (without avg_posts)	0.81	0.99	0.89	86
	spam (without avg_posts)	0.98	**0.70**	0.82	

adversarial hijacked accounts using Gilani-2017 [114] and cresci-rtbust-2019 [191] datasets. We used the RF algorithm in this experiments, and after training it with the training part of the datasets, we add 81 adversarial examples to the testing part of dataset and use it for evaluation. Table 4.13 shows that the accuracy of the RF trained on both datasets significantly drooped when removing *avg_posts*. The accuracy of the RF trained on Gilani 2017 dataset decreases by 19%, and it decreases by 9% in the RF trained on the Cresci-rtbust 2019. The overall detection accuracy of both datasets Cresci-rtbust 2019 and Gilani 2017 are low because the data distribution of these datasets and the testing datasets are different. However, the goal of using these datasets is to evaluate the effect of the adversarial hijacked accounts and the importance of the new feature using benchmark datasets.

These experiments show the importance of the *avg_posts* feature to supervised models using different ML algorithms for detecting the adversarial hijacked accounts. Also, they show the effect of the adversarial hijacked accounts on these models when trained with dataset that do not include adversarial examples. Training the models with adversarial examples and using the designed feature make the detection accuracy more stabled.

Table 4.13: Experiments results on Cresci-2019 and Gilani 2017 datasets

		precision	recall	F1 score	accuracy
gilani-2017	non_spam (with avg_posts)	0.56	0.86	0.68	**0.55**
	spam (with avg_posts)	0.48	0.16	0.24	
	non_spam (without avg_posts)	0.41	0.34	0.37	**0.36**
	spam (without avg_posts)	0.32	0.40	0.36	
cresci-2019	non_spam (with avg_posts)	0.78	0.56	0.66	**0.67**
	spam (with avg_posts)	0.6	0.8	0.68	
	non_spam (without avg_posts)	0.62	0.62	0.62	**0.58**
	spam (without avg_posts)	0.53	0.52	0.52	

4.4.3.1.3 Unsupervised Models In the above experiments, we evaluate the importance of the new designed feature using ML algorithms trained in a supervised manner. Here, we compare the importance of *avg_posts* in the detection of adversarial hijacked accounts using models trained in an unsupervised manner as some recent studies used unsupervised approaches to detect hijacked accounts [258, 149] and bots [191]. Although supervised approaches can detect spam with high accuracy, their detection accuracy drooped on detecting never seen data (i.e., Zero-day attacks). We employ anomaly detection auto-encoders (AEs) [217] to evaluate the effectiveness of *avg_posts*. AE is a type of dimensionalty reduction and feature projection techniques (e.g., PCA, TICA). We used an AE as some related studies show that they outperform other diminsionality reduction techniques in detecting compromised accounts in OSNs [258, 191, 149]. Three auto-encoders were built using different neural models, namely LSTM, BiLSTM, and Dense. We use Area under the ROC curve (AUC) as our evaluation metric. AE consists of two components: encoder and decoder. The encoder is used to extract principal components of the data and then reconstruct the original features from the principal components. The likelihood of a spam is measured by its reconstruction error. The encoder part of the three models trained on the non-spam samples only, and the decoder part of the models were used to classify the test dataset that includes non-spam and adversarial hijacked accounts. The three adopted AEs [6] consist of 4 layers: two encoders and two decoders. Similar to the above experiments, the collected dataset from Twitter was used for training and testing. Figure 4.23 shows that there is a considerable improvement in the performance of the three unsupervised models when using *avg_posts*. The AUC of the Dense-based AE improves by 12% when using *avg_posts*, wheres 1% and 2% improvements were recorded for BiLSTM and LSTM respectively.

These unsupervised models can be used to mitigate *causative attacks*, where adversaries try to contaminate the training dataset. Such approaches can ensure that the training dataset does not contain adversarial examples. However, the above experiments show that anomaly detection AE could not detect some of the identified adversarial hijacked accounts since our analysis show that some of the hijacked accounts have account age and number of posts similar to legitimate users' accounts. Future work will focus on preforming more analysis of these accounts.

4.4.3.1.4 Comparison Against Baselines Finally, we compare the detection accuracy of the adversarial hijacked accounts of our feature-based classifier against SOTA spam detectors. The

[6] https://github.com/syuecode/ai_software_dev/tree/master/AI

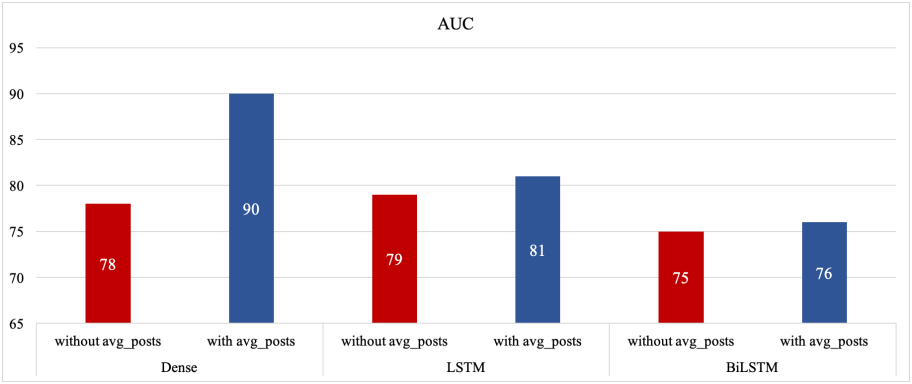


Figure 4.23: Performance of three auto-encoders using avg_posts.

following three baselines were chosen:

- *COMPA [95]*: is an algorithm that builds a behaviour profile for each account based on their tweets' features (e.g., language, and time) and then detects anomalies. We use features that can be derived from our dataset, such as time (hour of day), message text (language), message topic (hashtag), links in messages (URLs) and direct user interaction (mentions). The anomaly score for each account's features is calculated, and a new tweet's features are check against the accounts' threshold.

- *Nauta [202]*: similar to COMPA, this algorithm builds a behaviour profile for each user and checks weather the anomaly score of a new tweet's features are equal or higher than the defied threshold. However, they compute the anomaly score slightly different and use more features than COMPA. We use features that can be derived from our dataset, such as time (hour of day), message text (language), message topic (hashtag), links in messages (URLs), domain of a URL and number of tweets per day (Frequency).

- *Botometer [31]*: is a publicly-available web-page that allows checking if a Twitter account is bot or not. It performs account-based analyses to classify accounts in range of 0 to 5. zero being most human-like and five being the most bot-like.

As most of related studies in detecting hijacked accounts use users' behavioural-based approach, which requires analysing users' tweet history, we build a new testing dataset that contains 52 users' accounts. The goal of these experiments is to show how these campaigns can fool hijacked accounts detectors that rely on accounts tweets history. We choose hijacked accounts that have old account age with very few tweets on their accounts (i.e., less than 10) and those that have only few spam tweets. These accounts are hard to be detected by users' behavioural-based detectors since their profiles do not have enough variations. After extracting the tweets of the 52 accounts, we manually evaluate them using COMPA's and Nauta's algorithms. The reason for the manual evaluation is that these algorithms cannot build a behaviour profile for accounts containing less than 10 tweets as stated in [95]. For evaluating the *Botometer*, we check each account in the dataset and record the results. If the score is higher than 3.5, the account is classified as spam.

Table 4.14: Comparison between our meta feature-based classifier and SOTA detectors

Algorithms	Recall
COMPA	0.19
Nauta	0.36
Botometer	0.71
Our classifier	0.73

As the dataset contains only hijacked accounts, we compare the ability of the detectors to correctly find hijacked accounts (recall). The results in Table 4.14 show that our Meta feature-based classifier outperforms the three detectors in detecting the adversarial hijacked accounts. For a fair comparison, we only used six features (no_followers, no_favourites, no_listed, status, account_age, and avg_posts) in this experiment. Although the *Botometer* detector achieves a result that is very close to ours, it classifies 34 out of 52 accounts with score and remarks that the "score might be inaccurate". The reason these accounts could not be classified accurately is that they have not been active for a long time and do not have enough variations. Thus, if we considered these 34 accounts as miss-classified, the result of *Botometer* would be 0.25. A possible reason that *Botometer* was not able to detect most of the adversarial hijacked accounts with high confidence could be because these accounts can be considered as *Trolls* (i.e. software-assisted human accounts). Similar findings were observed in [132], where the authors stated that it is hard for Bot detection algorithms to detect Trolls. In addition, the COMPA and Nautu achieve very low detection accuracy because their efficiency depends on the number of tweets an account contains. Similar findings were discussed in [258].

4.4.3.2 Content-based Classifiers

This subsection consists of three parts: tweets' content, emoji and description classification. First, we compare the detection accuracy of the three text classifiers: Doc2vec [167], CapsuleNet [128], and BOW with TF-IDF [7] and the results are presented in Table 4.15. Then, we examine the robustness of the three classifiers to a character-level type of attack. Followed by evaluating the detection accuracy of the chosen text classifiers using tweets' textual description. Finally, the results of emoji-based classifier are presented.

4.4.3.2.1 Tweets' Content Classifier We have utilized gensim [8] for the implementation of doc2vec. Also, we adopted the CapsuleNet model proposed in [128]. The three classifiers were trained using the collected dataset from Twitter, which were split into training and testing datasets. The classifiers trained on the textual content of the collected dataset's tweets. Then, evaluated using the testing dataset. Table 4.15 shows that dec2vec achieves the best detection accuracy among the three classifiers. The rational explanation of this result is that dec2vec captures the meaning within embeddings [258].

An important step when designing an adversary-aware detector is to consider the robustness of the detector to adversarial examples. Since our analysis of the targeted campaigns reveals some

[7] https://github.com/susanli2016/NLP-with-Python
[8] https://radimrehurek.com/gensim/

Table 4.15: A comparison between different text classification models

Models	Precision	Recall	F1 Score	Accuracy
Doc2vec and RF [167]	99	98	99	99
TF-IDF + RF	99	97	98	99
CapsuleNet [128]	91	87	92	97

adversarial activities that are carried out by these campaigns (e.g., adding repeated characters or misspelt spam words), we extend these types of character-level manipulation and create adversarial test dataset. First, we manipulated the top 30 frequent words in spam tweets by replacing some characters with visually similar symbols or numbers. For example, Maca → M@ca, Forever → F0rever, or. ماكاا → ماكاا Then, we trained the classifiers using a clean version (i.e., does not contain adversarial examples) of the training dataset. Finally, we test the classifiers using the manipulated dataset. The results show that Dec2vec is the most robust classifier against the character-level attack. Figures 4.24 and 4.25 present the results of the three classifiers using cleaned and manipulated datasets. Based on these results, Dec2vec was chosen for the tweets' content classifier C_B.

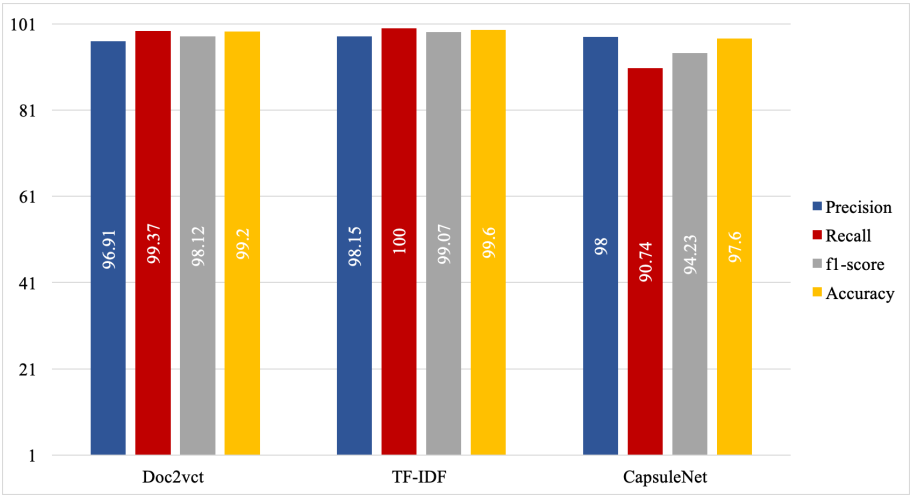


Figure 4.24: The results of the Three text classifiers using Clean Dataset

4.4.3.2.2 Tweets' Description Classifier One of the features of tweets posted under trending hashtags that often overlooked in the literature is an account's description. Tweets posted on trending hashtags consist of a combination of tweet and account features, and an account description is one of the features. To improve the detection accuracy of our detector and add another layer of defence, we analysis tweets' descriptions. We use Doc2vec classifier C_C for classifying tweets based on their descriptions. Although some of the tweets have an empty description, we found that the targeted campaigns use descriptions to mimic legitimate users' accounts. Our analysis shows that while a few, 12 out of 1990, non-spam tweets have empty description, 9 spam tweets have empty description. The Doc2vec with RF achieve 90% detection accuracy; Table 4.16 presents the classification report of our experiment using sklearn's library.

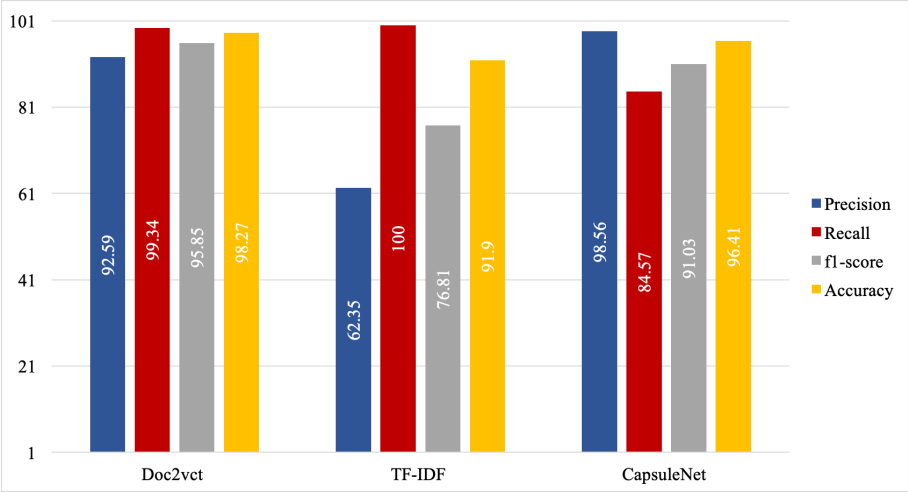


Figure 4.25: The results of the three text classifiers using perturbed dataset

Table 4.16: Detecting spam based on tweets' description using Doc2vec

	Precision	Rcall	F1 score	Support
Non-spam	0.89	0.99	0.94	607
spam	0.95	0.51	0.67	146
accuracy			0.9	753
macro avg	0.92	0.75	0.8	753
weighted avg	0.91	0.9	0.89	753

4.4.3.2.3 Tweets' Emoji Classifier Additionally, our analysis shows that spam and non-spam tweets use different emojis in their messages' content. Since the most frequent emojis used by the two classes were found different, we chose to use TF-idf for the classification task. First, we stripe emojis from tweets' content, and the final data distribution was 610 non-spam and 362 spam tweets. This shows that approximately 70% of spam tweets include emojis, while 30% of non-spam tweets use emojis. Then, we split the dataset into training and testing. Finally, we use TF-idf with RF for the classification C_D, and the results in Table 4.17 shows that our model can distinguish between the two classes with 98% detection accuracy.

Table 4.17: Results of emoji-based classifier

	Precision	Rcall	F1-score	Support
Non-spam	0.98	0.99	0.99	174
spam	0.99	0.97	0.98	114
accuracy			0.99	292
macro avg	0.99	0.98	0.99	292
weighted avg	0.99	0.99	0.99	292

4.4.3.3 Fuzzy Rule-based Classifier

To handle adversarial drift, two problems need to be considered: detecting possible adversarial drift and debugging/updating the detector. The proposed method for handling adversarial drift is a mix of active and passive approaches [186], in which we update the detector when the adversarial drift is detected (*i.e., active approach*) and when the classifiers disagree (*i.e., passive approach*). The methodology used for building this classifier was inspired by [228], which is one of the first studies that investigate the adversarial drift in streaming data. Table 4.18 presents all possible outcomes of the classifiers and the associated set of rules used by the FRB classifier. Based on the analysis of our dataset, we give the optional classifier C_C a higher score than C_D since we find that most of the tweets posted by accounts that has description. However, the sensitivity and weight of the classifiers can be updated if the disagreement between the classifiers increases. The FRB classifier will make its final decision based on the following three rules: 1) if both mandatory classifiers agree on an input class even if one or both optional disagree. 2) if one of the mandatory classifiers and both optional classifiers agree on an input class. 3) if one mandatory classifier agree with the optional classifier C_C. The output of the optional classifiers are considered only when the mandatory classifiers disagreed. These optional classifiers help overcoming the uncertainty and handling adversarial drift. Samples that the classifiers disagreed on, will be collected and used by the security analyst to update the classifiers.

Table 4.18: Fuzzy rules

Rule	Antecedent	Consequence
1	IF (C_A is 0) and (C_B is 0) and (C_C is 0) and (C_D is 0)	THEN (Y is 0)
2	IF (C_A is 0) and (C_B is 0) and (C_C is 1) and (C_D is 0)	THEN (Y is 0)
3	IF (C_A is 0) and (C_B is 0) and (C_C is 0) and (C_D is 1)	THEN (Y is 0)
4	IF (C_A is 0) and (C_B is 0) and (C_C is 1) and (C_D is 1)	THEN (Y is 0)
5	IF (C_A is 1) and (C_B is 0) and (C_C is 0) and (C_D is 0)	THEN (Y is 0)
6	IF (C_A is 1) and (C_B is 0) and (C_C is 1) and (C_D is 0)	THEN (Y is 1)
7	IF (C_A is 1) and (C_B is 0) and (C_C is 0) and (C_D is 1)	THEN (Y is 0)
8	IF (C_A is 1) and (C_B is 0) and (C_C is 1) and (C_D is 1)	THEN (Y is 1)
9	IF (C_A is 0) and (C_B is 1) and (C_C is 0) and (C_D is 0)	THEN (Y is 0)
10	IF (C_A is 0) and (C_B is 1) and (C_C is 1) and (C_D is 0)	THEN (Y is 1)
11	IF (C_A is 0) and (C_B is 1) and (C_C is 0) and (C_D is 1)	THEN (Y is 0)
12	IF (C_A is 0) and (C_B is 1) and (C_C is 1) and (C_D is 1)	THEN (Y is 1)
13	IF (C_A is 1) and (C_B is 1) and (C_C is 0) and (C_D is 0)	THEN (Y is 1)
14	IF (C_A is 1) and (C_B is 1) and (C_C is 1) and (C_D is 0)	THEN (Y is 1)
15	IF (C_A is 1) and (C_B is 1) and (C_C is 0) and (C_D is 1)	THEN (Y is 1)
16	IF (C_A is 1) and (C_B is 1) and (C_C is 1) and (C_D is 1)	THEN (Y is 1)

Where:

C_A: Output of statistical feature-based classifier.

C_B: Output of content-based classifier.

C_C: Output of description-based classifier.

C_D: Output of emoji-based classifier.

X: An input (Tweet).

Y: Class (0 or 1).

0: non-spam
1: spam.

4.4.3.3.1 Adversarial Drift Simulation To simulate the adversarial drift detection, we perform the following two steps:
Step 1) Detecting the adversarial drift:

1. Splitting the dataset into different chunks (D_1,..., D_n,...) where n is the number of chunks. The adversarial drift occurs between two points in time D_n^t and D_n^{t+1} where t is the time point. Each chunk contains a number of instances (i.e., tweets) $D_n^t = (x_1, ..., x_n, ...)$. if $P^{t(x,y)} \neq P^{t+1(x,y)}$, where $P^{t(x,y)}$ denotes the probability of data at a time point t, and y_n is the assigned class of input x_n.

2. Training our detector using clean dataset (i.e. not including adversarial examples)

3. Adding adversarial examples (hijacked accounts) to D_n^t with different percentages.

4. Evaluating our detector, which was trained on clean data, using testing datasets D_n^t.

5. The drift will be confirmed when the detection accuracy of the detector's classifiers dropped under the reference percentages.

In detail, we followed the methodology proposed in [228] for defining the reference percentages to which the predicted results of the classifiers were compared. The training dataset was used to find the expected accuracy for the classifiers. We uploaded the training dataset into WEKA and a 10-fold cross-validation was chosen as a test option. After repeating this process ten times, the learned expected behaviours of the classifiers were used for adversarial drift detection. Classifiers' sensitivity to drift can be controlled by modifying the reference percentages. The reference percentages of our detector are as follows: C_A: 95%, C_B: 96%, C_C: 84%, C_D: 89%.

After choosing the reference percentages (i.e. accepted drift) of our classifiers, now we are simulating the adversarial drift on our dataset. The number of samples that are considered as an indicative of the adversarial drift is depending on the classifiers used. The experiment was preformed using the meta feature-based classifier C_A. The adversarial attack scenario that we consider starts with a *probing attack*, where an adversary manipulates a few samples (i.e. tweets) and post them to learn from the deployed classifier's feedback. The next step is launching an adversarial attack, in which the adversary manipulates more samples to either evade detection or subvert the deployed classifier (i.e. Adversarial drift). To simulate the attack, we split the dataset into chunks (D_1,..., D_n,...). Each D_n^t denotes a set of samples that arrive at different time t. The training dataset consists of 400 non-spam and 100 spam tweets. Wheres, the testing datasets consist of 350 tweets that include different percentages of adversarial examples (*i.e., adversarial hijacked accounts*). The manipulation percentages start from 10% to 25%. The simulation of the adversarial drift is presented in Figure 4.26. The results show the probing attack starts at D_2, where the manipulation percentage is 10%, and the drift is detected at D_4, where the manipulation percentage is 14%.

After simulating the detection of adversarial drift, the next step is to debug/update the classifiers. Updating the detector requires obtaining labeled data, which connected with budget management

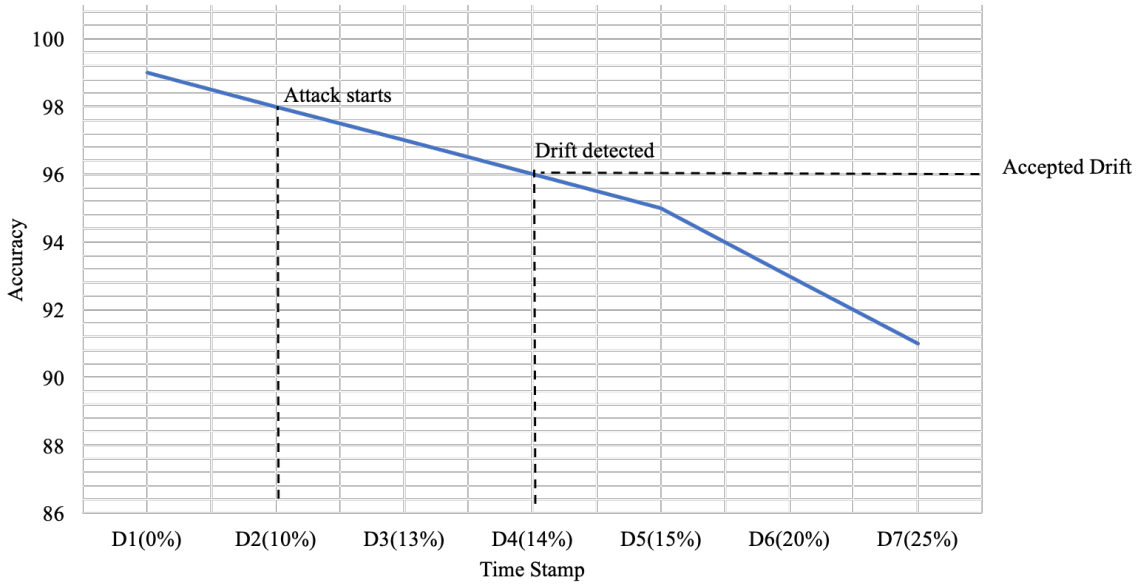


Figure 4.26: The simulation of adversarial drift.

(i.e., the monetary cost for paying the domain expert) and time (i.e., how quickly the expert can label instances) [153]. Different methods for collecting and labeling samples to update the deployed classifiers have proposed in the literature. Active learning focuses on choosing the most valuable data that need to be labeled, and has been widely used for solving this problem [152]. Several active learning methods are proposed to find the valuable samples, such as using uncertainty of a classifier [228], samples that best represent the concepts in distribution [105], or sliding windows [159]. Our proposed detector follows an active learning approach Query by Committee (QBC) [157] that finds the most valuable data to be used for updating the detector based on the disagreement between the classifiers. The QBC approach was first proposed for static active learning [109] and modified in [157] to be used for data stream. The labeling strategy we use is different from the one used by the adapted approach. We introduce our methodology for updating the detector in step 2.

Step 2) Updating and debugging the detector:

1. Samples that the classifiers (C_A, C_B, C_C, C_D) disagreed on (x_1, ..., x_n,...), where x_n is an input and n is the number of sample, will be collected. These samples will be labelled by the rule-based classifier using the fuzzy rules.

2. If the number of these samples reaches a certain level, they will be examined by the security analyst and used to evaluate the classifiers.

3. If the percentage accuracy of any of the classifiers dropped under the reference percentages, the collected samples will be used to update the detector.

4. Since the collected samples might not be sufficient for updating the classifiers, we will re-sample the collected samples by generating synthetic data [154].

The results in step 1 show that when manipulate 10% of the data, the detection accuracy drops. Thus, based on this result, in which we use the statistical classifier C_A, if the number of samples that classifiers disagreed on is higher than 10% of the arrived chunks, the samples need to be checked by the security analyst. Updating the deployed detector requires a certain amount of human work although it may be costly and time-consuming [152]. Also, Ksieniewicz et al., [159] stated that for some practical tasks (e.g. medical diagnosis) humans need to verify labeled data. Hence, we integrate HITL approach in the process of updating the adversary-aware detector since the targeted type of drift occurs as a result of adversarial attack. Once the drift is confirmed by the security analysts, the collected samples will be used for debugging. There are different methods for debugging the classifiers, and in this research we consider retraining as the method of debugging. In some cases, retraining the classifiers may not sufficient and designing a new feature or using different ML algorithm is needed. If the collected samples were not sufficient for updating the classifiers, data re-sampling need to be considered. Data re-sampling can be categorized into two groups: *undersampling*, where the majority class needs to be downsized, or *oversampling*, in which the minority class needs to be amplified. There are several oversampling methods that proposed in the literature, such as Random Over Sampling (ROS), Mahalanobis Distance-based Over-sampling technique (MDO) [1] and Synthetic Minority Over-sampling Technique (SMOTE) [59]. However, these methods have some limitations. The ability of ROS and MDO to protect minority class samples is limited, whereas SMOTE suffers from over-generalization [292]. To overcome drawbacks of SMOTE, some extensions have been proposed. For example, ADASYN [125], RAMO [65], MWMOTE [25], and several SMOTE variants (e.g., Borderline-SMOTE (SMOTE-B) and Safe-level-SMOTE (SL-SMOTE). The simplicity, computational efficiency, and superior performance of SMOTE over other oversampling methods have made it the most frequently used technique [187, 281, 116]. Thus, SMOTE was chosen to generate new artificial samples by replicating pre-existing ones in this study.

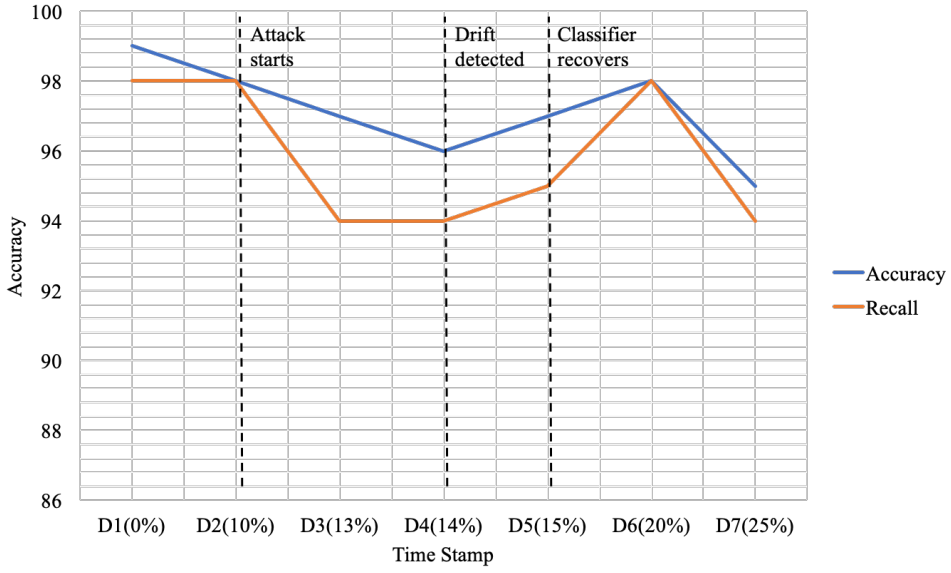


Figure 4.27: We use the detected adversarial examples (D_4) to update the classifier

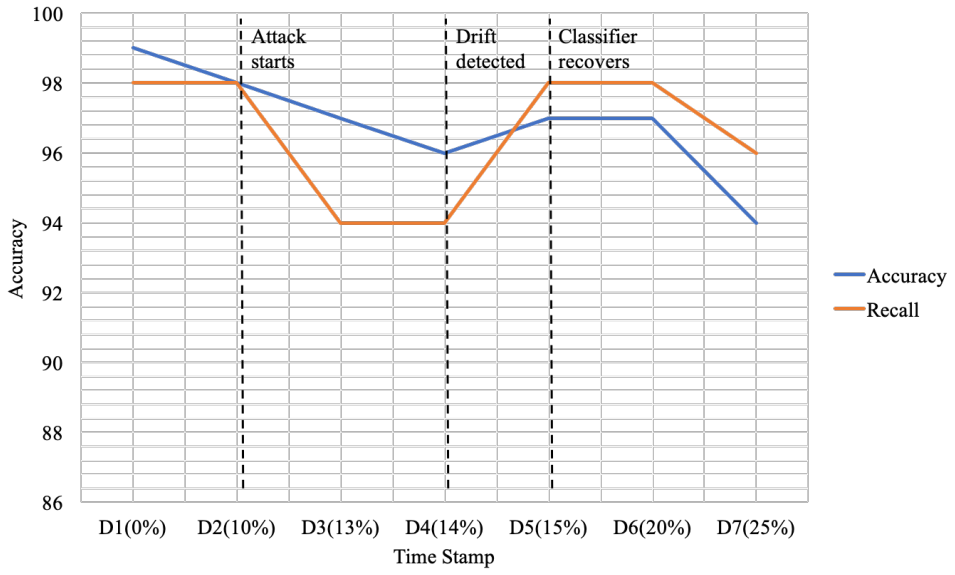


Figure 4.28: We use the detected adversarial examples (D_4) and the oversampling technique SMOTE to update the classifier

In Figures 4.27 and 4.28 we compare the accuracy and recall of two retraining methods used by the classifier C_A to handle the adversarial drift. We use the same setting for simulating the adversarial drift as in the previous experiment. In Figure 4.27, after the drift is detected at $D_4(14\%)$, we used D_4 to retrain the classifier. We considered the classification accuracy and recall since the adversarial drift occurs as a result of manipulating the malicious samples only [228]. In Figure 4.28, we retrain the classifier using SMOTE from Imbalanced-Learn Library[9]. We over-sample D_4 and update the classifier. The results show that using SMOTE makes the recall more stable than updating the classifier using the detected adversarial drift's samples.

4.5 Discussion

Health-related ads campaigns remain an ongoing issue in Arabic trending hashtags. These campaigns not only spread malicious content, they also affect users' experience. Regularly, users visit trending hashtags, a primary feature on Twitter, to access news and share their thoughts about topics that they are interested in. However, spam campaigns flood trending hashtags with spam messages, which makes it difficult for users to read tweets and communicate with each other. This kind of spamming activity is called *hashtag hijacking*. The analysis of the hashtag in Section 4.3.1.1 shows how these campaigns have a negative effect on users' experience. Most of the spam tweets posted by these campaigns are detected by Twitter after some time. This delay in detection makes these spam tweet spread very quickly since the health-related ads campaigns retweet them by using bots, as has been shown in the previous sections. Also, some of these spam tweets last over six hours without detection, which is a long time given the fact that these tweets show clear signs of being spam.

[9]https://github.com/scikit-learn-contrib/imbalanced-learn

Figure 4.29 provides an example of what appears to be a spam tweet that had not been detected by Twitter for over two weeks. It might be argued that these spam campaigns are not dangerous. In a previous study, we have shown how these activities can affect the deployed spam detection system and lead to more harmful attacks [142]. These campaigns also use spam tweets that contain images with no text to bypass traditional detection system that rely on tweets' content [133]. Moreover, the analysis in this chapter shows that these campaigns use hijacked accounts.

Figure 4.29: An example of a spam tweet remain undetected for over two weeks.

Existing studies of spam campaign analysis focus on the model's detection accuracy, but only a limited number of studies have considered the robustness, adaptability and human-interpretability of the detection models. Recent research that studied the vulnerability of ML-based models (*i.e., Adversarial ML*) shows that, when designing an ML-based model for cybersecurity systems, the presence of adversaries and how the model can evolve over time need to be considered at the design stage [43]. One of the characteristics that makes ML-based models vulnerable is overemphasizing some features to discriminate between classes. Adversaries can exploit this vulnerability by manipulating the most influential features [193]. Finding this type of feature in OSNs is not difficult, as adversaries can launch an exploratory attack against the deployed ML-based model [142]. Hence, an important goal of this chapter was to find features that are either hard or expensive to manipulate. Our experimental results of section 4.4 show that the new designed feature is one of the most important features to the selected classifier (RF). Using this feature can improve the detector's robustness against adversarial activities as it is based on unchangeable temporal patterns of Twitter accounts. Also, the importance of the new feature in detecting the adversarial hijacked accounts was demonstrated using supervised and unsupervised classifiers.

The analysis in this chapter focused on health-related ads campaigns; however, the new designed feature is capable of detecting any hijacked account that has characteristics similar to those discussed earlier. Different spam campaigns using similar hijacked accounts were found, such as pornography and religious propaganda campaigns. Figure 4.30 shows a tweet posted by a hijacked account that was created on 2012 and has only 6 tweets in its timeline. This tweets belong to a religious

Figure 4.30: An example of a spam tweet posted by hijacked accounts used by a religious propaganda campaign

propaganda campaign. Also, one of the things that makes this feature so effective in detecting these accounts, is that it does not rely on an account's tweeting history. It was designed to detect these adversarial hijacked accounts in hashtags, where the analysis of tweets' characteristics need to be done very quickly. Thus, we designed our feature to consider the gap between the creation date (*account_age*) and the number of messages (*status*) for tweets as the results of our analysis show that the gap for the adversarial hijacked accounts is abnormal. Even if adversaries try to post more messages per day to evade detection, it is difficult to reduce the gap between the two features before being detected. Additionally, the proposed feature can be used to reduce the computational cost of existing hijacked accounts' detectors that utilized accounts behavioural-based approach. These hijacked accounts detectors can use the proposed feature as an indication that an account is hijacked and then they can confirm that by examine the account tweeting history (*RQ2 answer*). Although this chapter focuses on spam campaigns in Arabic trending hashtags, the new feature can detect hijacked accounts in any language. Another advantage is that, even if adversaries craft tweets containing only an image or video to fool the detection model [100, 139], the proposed adversary-

aware detector can detect these, as it considers the statistical features of tweets and the textual content. On the other hand, there is a possibility that users who are not especially active may have low average number of posts, similar to the identified hijacked accounts. Also, some hijacked accounts have a total number of posts over the account's lifetime that is similar to a legitimate account (see Figure 4.31). However, as the developed detector does not make predictions solely based on the new feature, it can distinguish between non-spam and spam taking other features (e.g. content-based) into account. The proposed adversary-aware detector was developed to detect possible adversarial drift that may occur as a result of adversarial attack (*RQ3 answer*). Also, the proposed detector can be generalized to other social media platforms for detecting adversarial examples since most of these platforms have similar features. For example, both Twitter and Facebook messages have account age and number of messages. Also, both platforms allow users to share text, images and videos.

Figure 4.31: A hijacked account created in 2016 with more than 49k tweets. These features are similar to legitimate account features

The study of health-related ads campaigns demonstrates that designing a detection system for spam campaigns rather than accounts run by bots helps designing a more divers spam detector. Our analysis shows that the targeted campaigns use different spamming tactics (e.g. bots, trolls, and hijacked accounts), which can not be detected efficiently by bots detection algorithms. Early detection of hijacked accounts used in these campaigns is important. The dataset analysis shows that these accounts are used to speed up the spreading of spam tweets by retweeting, mentioning, commenting, and liking. The longer these hijacked accounts remain undetected, the harder it is to discriminate between them and regular accounts. These findings answer *RQ1* that the targeted campaigns use different spamming activities to fool Twitter spam detectors. Additionally, the results of the experiments show that the identified hijacked accounts could not be detected by account's behavioural-based detectors. These hijacked accounts use inactive accounts have different characteristics than the hijacked accounts discussed in the literature.

There are different adversarial attacks that can be launched against spam detectors in general. Although spammers can adopt powerful AI-driven techniques for automatically generating texts, there are some constraints that may render them from using some of these techniques. For example the targeted campaigns' goal is to advertise male enhancement products, so their text manipulation method will focus on a list of words. Moreover, the adversaries need to consider human perception and character-level perturbation is hard to be crafted. Thus, we only consider the most possible realistic text manipulation, where adversaries change few characters in the most frequent words.

4.6 Summary

Motivated by the spread of untrustworthy healthcare advertisements in Arabic trending hashtags, we have carried out a first study of this type of spamming campaign. The analysis of health-related ads campaigns on Twitter uncovered some important characteristics that can help detecting such campaigns. These campaigns were found to widely use unique hijacked accounts, which are old and/or inactive accounts with few posts. This observation led to the design of a new robust feature that can detect hijacked accounts on Twitter hashtags. Several experiments were performed to evaluate the importance of the new feature to the classification task. The new feature is capable of improving the detection of spam tweets in any language, as it considers an account's temporal patterns. The study of the targeted campaigns shows that these campaigns can not be detected efficiently by bot detectors as they use bots, trolls, and hijacked accounts. Our analysis of the collected dataset shows that 40% of the hijacked accounts were still active, whereas 13% of spam bots were found non-suspended. Also, we find that the targeted campaigns hijack inactive accounts (*adversarial hijacked accounts*) to fool users' behavioural-based detectors. Excessive experiments on the collected datadets show that our adversary-aware detector out preforms bots and hijacked accounts detectors. Additionally, the developed detector, which consists of MCS and a FRB classifier, was designed to be robust the identified adversarial examples, adaptable to handle adversarial drift and interpretable to evolve over time.

The aim of this study was to simulate the research community to focus on designing adversary-aware detection systems that are robust, adaptable and interpretable. Although the analysis focused on spam campaigns in Arabic trending hashtags, as mentioned, the new designed feature can detect hijacked accounts regardless of the language used. Finally, achieving a high detection accuracy was not the main goal of this research, as the literature proves that, with enough data, it is not difficult to achieve high accuracy. Rather, our main focus was to develop adversary-aware spam detector keeping into accounts three key points: the robustness to the identified adversarial examples, adaptability and interpertability to handle adversarial drift (i.e., to ensure the detector can evolve over time).

Chapter 5

Spam Images Detection: Improving Robustness of OCR-based Detectors by Fine-tuning with Adversarial Spam Images

The third contribution of the thesis is presented in this chapter. Similar to the previous chapter, we continue analysing the targeted spam campaigns in Twitter Arabic hashtags. After identifying that these spam campaigns use a large number of images with embedded malicious text as adversarial examples, we proposed an approach for developing an adversary-aware Optical Character Recognition (OCR)-based detector that is, robust, adaptable and interpretable. We improve the robustness of existing OCR-based detectors using a dataset of adversarial spam images collected from Twitter. Also, we design an adaptable and interpretable text classification model as part of the OCR-based detectors to ensure that our adversary-aware, OCR-based detector can evolve over time. This chapter includes and expands on my papers published in:

- *Imam, N. Vassilakis, V., 2019. An Approach for Detecting Image Spam in OSNs. Proceedings of the Conference for Truth and Trust Online 2019.*
- *Imam, N. and Vassilakis, V., 2019, September. Detecting Spam Images with Embedded Arabic Text in Twitter. In 2019 International Conference on Document Analysis and Recognition Workshops(ICDARW) (Vol. 6, pp. 1-6). IEEE*

5.1 Introduction

Existing spam detectors that use text-based, statistic info-based and even graph-based features can easily be fooled by spam images, where an adversary inserts text inside an image. A recent survey conducted by [276] presented the pros and cons of these detectors and suggested of developing a comprehensive model to improve detection performance. Spam images have been replaced by URL-based spam, as the latter requires a lower email size and is able to send more messages. However, the volume of images being shared in Online Social Networks (OSNs) has been growing, partly due to the increase in network bandwidth. Processing large numbers of images and the retrieval of textual content from images are challenges in OSNs. Although some of the OSNs' platforms, such as Twitter has provided Muting options to enable users to block messages (i.e., tweets) contain particular words or hashtags, these Muting options can not block spam image. Figure 5.1 shows some examples of spam images found in Twitter. Setting your Twitter account to mute words, such as 18+, or MULTI MACA, will not block the examples in Figure 5.1. Thus, a solution that considers extracting text from images is therefore needed.

Figure 5.1: Examples of spam image.

Additionally, the number of spam images with embedded Arabic text in trending hashtags was found to be high, the reasons for which are beyond the scope of this study. However, it was found that there are very active spam campaigns spreading advertisements for untrustworthy drugs, for example weight loss drugs, Viagra, and hair treatment drugs, targeting Arabic-speaking users. Figure 5.2 shows a spam tweet that contains an image with embedded Arabic text. This kind of spam tweet should be detected easily as the image contains a very common spam word "Viagra". However, as the spam word is embedded in an image, the spam massage bypasses the deployed detector. Such examples show that there is a need for developing spam detectors capable of recognizing Arabic text in images. There are numerous techniques developed to detect and recognize scene text [290, 234]. However, very limited studies focus on developing models for detecting and recognizing artificial text overlaid on OSNs' images, especially, images with embedded Arabic text. The authors in [285] listed five special characteristics that distinguished Arabic text from Latin text. These characteristics make detecting and recognizing Arabic text more challenging. Recently, a few research papers in the literature proposed to recognize Arabic text in video images and printed documents [209, 28, 56, 285].

Recent works have adopted OCR systems to developing spam images detect for OSNs [49, 283].

Figure 5.2: An image with embedded spam word

According to [239] the number of images uploaded to Facebook is now in the hundreds of millions. Spammers take advantage of images on OSNs by inserting malicious content into images to evade detection. One technology that enables both handwritten and printed text to be extracted from images is OCR. Although OCR has shown some weakness in the past, seminal work by [168] and other advances in deep learning for object recognition tasks (e.g., Connectionist Temporal Classification (CTC), and Convolutional Neural Network (CNN) has helped to overcome some of these drawbacks [118] [85]. These works use Deep Neural Network (DNN) as a feature extractor, which allows use of variable-size image inputs [241]. The OCR systems are used for extracting text embedded on images, and then a text classifier is used to classify images.

Common methodologies used in the literature for classifying extracted text from images are based on NLP techniques, such as Word2vector and Bag-of-Word (BoW), or Multi- layer perceptron (MLP). However, recent studies have shown that these techniques are vulnerable to malicious activities, enabling an adversary to mislead deployed models. Adversaries can easily fool NLP models by adding, removing or replacing words [218]. Also, as cited in [97], the lack of robustness to morphological variation or spelling mistakes can be exploited as a blind spot of NLP techniques. These examples show the distinction between human and machine learning models. Human intervention is important to defend against adversarial attacks.

Figure 5.3: Images with embedded manipulated text

This chapter presents the first study of images with embedded malicious content that posted

by the ongoing health-related spam campaigns on Twitter Arabic hashtags. Specifically, we developed an adversary-aware OCR-based detector, leveraging two key observations about these spam campaigns' messages: (1) they embedded spam words into images to obfuscate traditional spam detectors (see Figure 5.2), (2) and they purposely manipulate the spam words to fool the deployed detection system (see Figure 5.3). Our detector exploit these observations by utilizing an OCR for extracting English/Arabic text from spam images and than the text is classified as spam or non-spam by a text classification model. The developed detector is designed to be robust to the identified Adversarial Spam Images, adaptable and interpretable to detect new or modified embedded words (i.e., evolving attacks). A dataset of images with embedded English/Arabic text (adversarial spam images) were collected from Twitter to fine-tune and improve the robustness of the adopted text detection (PixelLink [87]) part of the OCR. Also, a large dataset of cropped synthetic and Twitter images was created to build the text recognition (CRNN [234]) part of the OCR. Various experiments which altered different parameters of the adopted OCR were performed to detect image spam with Arabic text on Twitter. Finally, we proposed a black/ white list with human assistance model for classifying extracted words. This simple text classification model is designed to ensure that the developed detector can evolve over time by detecting possible text manipulation attacks. Our goals in this chapter are as follows:

- Evaluating the performance of an existing OCR system to the identified adversarial spam images.
- Improving the performance of the existing OCR system by fine-tuning it with adversarial spam images.
- Developing an adversary-aware OCR-based detector considering the three key points.

The remainder of this chapter is structured as follows: Section 5.2 provides an overview of previous researches. Section 5.3 describes the methodology used for developing our adversary-aware OCR-based detector of spam image in Twitter; experimentation and results are presented in Section 5.4. The summary of this chapter is presented in Section 5.5.

5.2 Related Work

There are two types of image spam detection methods: content-based and characteristics-based. The first type attempts to extract the text in a spam image and then make detection decisions. Similarly, it would follow the same process with non-image spam. On the other hand, the second type attempts to detect image spam based on the characteristics of image files. The research in this chapter pursues the first approach, but first, a revision of some relevant related work for characteristics-based approaches is going to be discussed. In [126] the authors develop a comprehensive model for detecting different types of spam in OSNs, including spam image. The developed model can detect images based on the following features: Colour and Edge Directivity Descriptor (CEDD), Gabor features, edge histograms and the Scale Invariant Feature Transform (SIFT). Also, [15] proposed an approach for detecting spam images based on a set of images' characteristics. The proposed approach extracts 21 features, such as colour, edges, comp, noise, etc., for training a linear SVM

classifier. Two different datasets were used: a standard dataset and an improved dataset (more challenging). Although the prediction accuracy of the standard dataset reaches 95%, the model was not capable of distinguishing between spam and non-spam images accurately when applied to an improved dataset.

On the other hand, a recent study [49] developed images content-based detector that uses an OCR system that detects and recognizes text in images uploaded to Facebook. The system, called Rosetta, consists of two models: text detection and text recognition. The text detection model uses Fast-RCNN to perform word detection. It detects the locations of words in an image and produces words surrounded by bounding boxes. Then, for each detected box, a fully-convolutional model, referred to as CTC, is used to recognize text. The recognition model predicts the most likely character at each detected box in the image. For experiments, different datasets were used; a synthetic dataset was used for pre-training and COCO-text, and human rated datasets were then used for fine-tuning the models. The developed models were implemented in Detectron, an open-source software used for object detection research. Also, authors in [283] developed an image content-based model called Malena, which can detect different types of spam including images carrying text, number, or QR code in Chinese social networks (Baidu Tieba and Sina Weibo). The authors use an off-the-shelf tool, PixelLink [87] for detecting text in images.

Several works have proposed Arabic OCR systems for printed and handwritten documents [56], and a few works developed for Arabic text in news video [285]. In [56], the authors developed Arabic OCR for handwritten documents using K-means-based approach for detecting text in a document. After removing noise from the document, they used a Hidden Markov Models (HMM)-based approach for text recognition. The results show an improvement in recognizing handwritten text. Authors in [254] proposed a robust classification framework for Arabic Scene Text Characters (STC). Bag of Feature (BoF) was used for character recognition. Also, another approach proposed in [3] for recognizing Arabic scene text using ConvNets. English Arabic Scene Text (EAST) dataset was used for training the model after it had been pre-processed. A small subset of collected images with Arabic text were used to evaluate the ConvNets, and the results were encouraging.

Attacks against text classification models have been widely studied in the literature. In [172] spaces between words are replaced with special characters, such as hyphens or asterisks to fool word2vector models. Also, [97] investigated the robustness of SOTA Deep Learning models against visual attacks, in which an adversary exchanges some characters with alternative visually similar ones (i.e. V1agra). These attacks are common in OSNs as they do not require any knowledge about the deployed model, nor any linguistic knowledge or and human understanding.

In contrast to existing spam image detectors [49, 283], our adversary-aware OCR-based detector of spam images is designed to be robust, adaptable and interpretable. These three key points for designing an adversary-aware OCR-based detector have not been considered in related studies. Hence, existing spam image detectors may not be reliable since they either adversary-agnostic or only consider some of the aspects of adversarial environments. Although the detector *Malena*, which has been proposed in [283], was designed to be robust to perturbed images (e.g., blur, or noise), it was not designed to evolve over time. Additionally, few studies use OCR system to understand text in scene text characters [254], scanned documents [56], or videos [285]. However, to our knowledge, there is no study that uses OCR for detecting images with embedded Arabic text in Twitter hashtags.

5.3 Our Method

The methodology used for building our adversary-aware OCR-based detector follows the methodologies used in related studies [49, 283]. However, our detector was designed taking into account the presence of an active adversary and evolving attacks. It involves three independent steps: text detection, recognition, and classification. In the first step, text regions in the spam image are detected. In the second step, the detected text regions are recognized and saved as text file. In the last step, the recognized words are classified as either spam or non-spam. Figure 5.4 illustrates the architecture of the developed adversary-aware OCR-based detector.

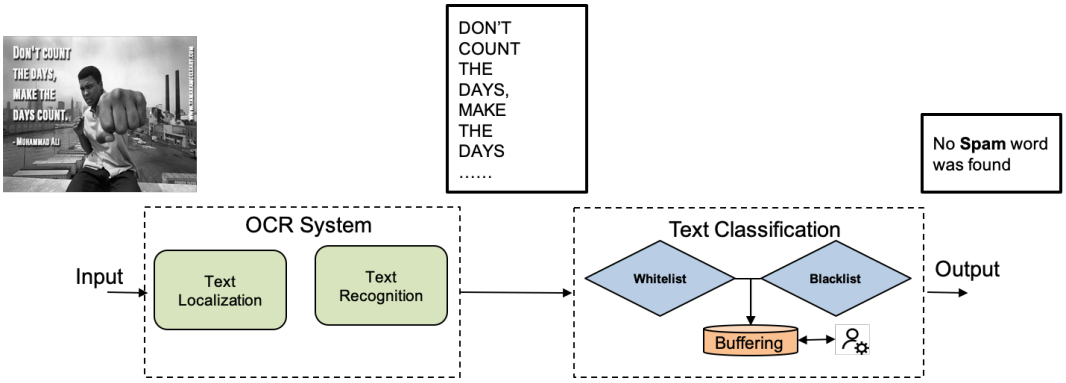


Figure 5.4: Overall architecture of our adversary-aware OCR-based detector. It consists of three models: text detection based on PixelLink, word recognition using CRNN, and a proposed text classification model.

5.3.1 Text Detection Model

The model used for text detection/ localization is called PixelLink [87]. It detects tasks through instance segmentation, in which pixels within the same text instance are linked together. Then, detected text (i.e bounding boxes) is directly extracted from the segmentation result, without performing location regression. The backbone of PixelLink uses VGG16 [238] as the feature extractor, with fully connected layers. The model has been widely used in the literature [180, 173].

In detail, PixelLink uses a neural network to perform predictions on pixels (text/non-text) and links. First, it finds how likely a pixel is part of a text instance, and then determines if adjacent two positive pixels can be linked together and are related to the same instance. Second, it joins these linked text pixels to detect the content of the text via text instance segmentation [283]. A pre-trained model, trained on ICDAR 2013 [148], 2015 [147] and MSRA-TD500(TD500) benchmarked datasets was used for transfer learning. The pre-trained model achieves an 84.5 F-meanand and 83.6 recall on ICDAR 2013 test dataset. ICDAR 2017 and the developed Twitter datasets were used for fine-tuning the model. Detection accuracy results are discussed in the following sections.

5.3.2 Text Recognition Model

An approach based on Convolutional Recurrent Neural Network (CRNN) proposed in [235] was adopted. The CRNN model is a combination of Deep Convolutional Neural Network (DCNN) and Recurrent Neural Network (RNN). The architecture of CRNN consists of three components: Convolutional Layers, Recurrent Layers, and Transcription Layer.

Convolutional layers are constructed to extract sequential features from an input image. The input image is divided into columns from left to right. Each feature map column corresponds to a rectangular region of input image. Thus, each vector feature of a feature sequence is considered as a descriptor of that rectangular region. Then, the recurrent network is used to make predictions for the output of the convolutional layers, which is a set of feature sequence frames. A label distribution for each frame in the feature sequence x is predicted by the recurrent layers. One of the advantages of recurrent layers is that RNN can capture contextual information within a feature sequence. As the traditional RNN suffers from a vanishing gradient problem, Long-Short Term Memory (LSTM) was used. LSTM consists of memory cells to store the past context, and input/output gates to store context for a long period of time. The third component of LSTM is forget gates, which are used to clear the memory cell. A deep bidirectional LSTM, which consists of forward and backward LSTMs, was used. The transcript layer translates the per-frame predictions of the recurrent layers into a label sequence. It finds the label sequence that has the highest probability conditioned among the pre-frame predictions. This layer uses CTC for the conditional probability task. There are two transcription modes that can be used: lexicon-free and lexicon-based transcripts. In lexicon-free mode, the probability of sequences are taken as the predictions. However, in lexicon-based mode, the probability are associated with lexicon, which is a spell-checking dictionary.

5.3.3 Text Classification Model

A simple approach is used to classify the extracted text. To ensure that our adversary-aware OCR-based detector can evolve over time to emerging attacks, we design our detector to capture new manipulated embedded text and use these adversarial text to update the detector. As the output of the text recognition is a set of words, a blacklist of spam words is used. We have adopted a blacklist that was created in 2011 and collected from Wordpress comments [1]. Additionally, a Natural Language Toolkit (NLTK) package was used to build a whitelist of non-spam words, which helps to detect a new crafted word that might be used by an adversary. Consequentially, if a spam word is detected, the model will notify a user that this image is a spam image, or if a new word that cannot be found in neither the blacklist nor the whitelist is detected, the new word will be stored to be evaluated by an oracle (i.e., security analyst) and the user will be notified that the image may contain sensitive content. The security analyst's tasks are examining new detected words and updating the lists. A flowchart for the text classification model is illustrated in Figure 5.4.

[1] https://github.com/splorp/wordpress-comment-blacklist

5.4 Experimental Results

Three steps have been preformed to build and evaluate our adversary-aware OCR-based detector. First, several synthetic and real-world datasets have been build for fine-tuning and testing parts of the adopted OCR system. After training the OCR-based detector's parts individually, the performance was evaluated and compared with baselines.

5.4.1 Datasets

Different datasets were used for training and evaluating the adversary-aware OCR-based detector. Each part of the detector was trained independently using different dataset unlike existing OCR-based detectors that are trained in an end-to-end fashion. This training method adds some robustness to the detector against training data poisoning (i.e., *Causative attacks*) types of adversarial attacks as it segmented the detectors' models. Description of datasets used for training and testing are provided as follows:

- **ICDAR2017:** A public dataset built for ICDAR2017 Competition on Multi-lingual scene text detection and script identification [203] was used for training the text detection model. The dataset is a collection of natural scene images with embedded text. It consists of 18,000 images containing text, such as street signs, advertisement boards and shops names, for 9 languages and symbols (see Figure 5.5). This enables it to be used as a benchmark for testing algorithms' ability to distinguish different scripts in images. Although it does not match images found in Twitter exactly, it resembles some of Twitter's images' characteristics, such as languages, fonts, natural scene backgrounds, and text locations and directions.

- **Twitter image dataset:** A small dataset of 300 images was collected and manually annotated to fine-tune the text detection model as there was no publicly available dataset for images with embedded Arabic text. The dataset was published for future use by researchers [135]. Figure 5.5 presents some examples of images found in Twitter hashtags. Twitter's images have a combination of the following characteristics: variation of aspect ratio, multi-oriented, curved text, variation of fonts, and multilingual text. Also, the two groups of images in Figure 5.5 show the difference between ICDAR 2017 dataset's images and images found on Twitter.

Figure 5.5: Examples of images used for training the text detection model

- **A synthetic word dataset:** It consists of 90 thousand English words, and it was used for training the adopted text recognition model [143].
- **Arabic synthetic dataset** A framework for generating synthetic datasets that do not require human-labelling were adopted. The framework was adjusted to generate cropped images with embedded Arabic text. The labels were generated from an Arabic words corpus, which consists of 15 thousand words [99]. This datasets was used for training the adopted text recognition model.
- **Arabic-Text-in-Video (AcTiV):** This was the first publicly accessible dataset built to assess the performance of different Arabic Video OCR systems. (see Figure 5.6). The dataset was presented at the ICDAR 2015 conference [284]. It was pre-processed as it consists of Arabic cropped sentences, while the adopted text recognition model requires cropped images with a single word featured. Also, the ground truths types of file were converted from XML to txt.

Figure 5.6: Samples of AcTiV

- **Twitter cropped images:** This dataset was built by collecting cropped images from Twitter to improve the robustness of the adopted text recognition model (see Figure 5.7). The collected images have the following characteristics: variability of text colours, fonts, sizes, position, and complex backgrounds.

Figure 5.7: Samples of Twitter cropped images

- **Wordprees comments and SMS spam datasets:** These two datasets were used to create both the blacklist and white-list for building the text classification model. The Wordpress dataset contains 36,000 phrases, patterns, and keywords. The SMS spam dataset [12], which is commonly used in literature for building spam detection models, contains a set of 5,574 SMS massages classified as spam or ham. The list of spam words was used as a blacklist, while Ham

messages corpus was extracted and Wikicorpus [2], which widely used in NLP applications, were used to build the white-list. Both lists will be updated regularly as spam words used in OSNs might be different.

5.4.2 Training

Some recent studies proposed end-to-end OCR-based models that use a single dataest for training both text detection and recognition models. However, in this research, the developed adversary-aware detector is trained in a two-step fashion, where each part of the detector is trained separately. This approach has some advantages, including the ability to update a single part of the model when it is needed. Most importantly, this approach ensures that if part of the model is been compromised, other pats of the model would not be affected.

5.4.3 Performance Evaluation

Three experiments were preformed to evaluate and compare results of the developed adversary-aware OCR-based detector with two SOTA OCR systems. Since we train our detector's models independently, we evaluate the performance of each separately. We follow the evaluation method used in related studies. Different evaluation metrics were used for evaluating the models. Metrics used for evaluation were adopted from ICDAR 2015 competition [147].

5.4.3.0.1 Text detection model. First a dataset of images with embedded text was collected from Twitter to test the text detection model as there is not an OSNs bench-marked dataset. 300 images with embedded text were collected from Twitter, and the dataset was split into 200 training, 50 validation and 50 test images. These Twitter datasets were manually labeled using a public tool called labelImg [3] and published at Mendeley [135]. These datasets contain *adversarial spam images* (i.e., spam images with noisier background and text distributed allover the image).

The goal of this experiment is to evaluate and improve the robustness of the adopted text detection model. We used two text detection models trained on ICDAR 2013 + 2015, and ICDAR 2017 datasets. Then, we fine-tuned the best model with Twitter datasets. Table 5.1 shows the evaluation results of three models: ICDAR 2013 + 2015, ICDAR 2017, and ICDAR 2017 + Twitter. The model fine-tuned by our *adversarial spam images* shows an overall improvement in the performance of the detection better than the other two models.

Metrics used for evaluating the models are: Precision, Recall, and F-measure (F1 score), that is the harmonic mean of precision and recall. These metrics were defined in the ICDAR 2013-2015 challenge. After annotating the test dataset collected from Twitter, the ground truth were compared with the result of each model.

5.4.3.0.2 Text recognition model. The goals of this part of the experiments are to improve the robustness of the adopted text recognition model using datasets collected from Twitter and to compare the results of our OCR system against SOTA OCR systems. The dataset collected from

[2] https://dumps.wikimedia.org/enwiki/latest/
[3] https://github.com/tzutalin/labelImg

Table 5.1: Evaluation of the text detection model using models fine-tuned by three different datasets.

No.	Model	precision	recall	f1 score
1	ICDAR 2013 + 2015	0.65	0.60	0.62
2	ICDAR 2017	0.78	0.65	0.71
3	ICDAR 2017 + Twitter	0.79	0.68	0.73

Twitter to test the text detection models was cropped to be used for evaluating the text recognition model. The total number of samples used to test this model is 120 images with embedded Arabic and English text. The performance of the model in recognizing Arabic and English text were compered after training it with synthetic and real-word datasets. Table 5.2 presents the results of the text recognition model using Twitter test dataset.

The code adopted for evaluating the trained text recognition model were built by D. Karatzas et al. [147] in Incidental Scene Text 2015 competition. The metrics used for evaluation are: (CRW) Correctly Recognized Words, and (TED) Total Edit distance[4]. Table 5.2 shows the CRW for English samples is higher than the Arabic one, which means that the model can recognize English text better than Arabic text. One of the reasons is that the number and quality of English language samples used for training the model is higher than the Arabic ones. The number of English samples in the dataset built in [143] is 90 thousand high than the created Arabic samples.

Table 5.2: Evaluation of the text recognition model using Twitter test dataset.

No.	Language	No. of Samples	TED	CRW
1	Arabic	102	166.0	0.460
2	English	49	51.0	0.571
3	Arabic + English	121	199.0	0.305

In the second part of the experiments, we compare the recognition accuracy of our developed OCR system against SOTA OCR systems (Google Cloud Vision OCR and OCR.space). These OCRs were chosen as they have been used as baselines in a related study [283]. Table 5.3 shows the results of the three OCRs tested on the collected dataset from Twitter. Our OCR outpreforms the two OCRs in recognising Arabic and English text embedded in images uploaded into Twitter.

Table 5.3: Evaluation Results of the developed OCR model and two SOTA OCR models

Language	Metrics	Our OCR Model	Google Vision	OCR.space
Arabic	CRW	0.460	0.333	0.205
	TED	115.0	238.0	297.0
English	CRW	0.571	0.510	0.244
	TED	51.0	61.0	141.0

5.4.3.0.3 Text classification model. As this model is highly dependent on the output of the text detection/recognition models, no evaluation has been carried out for this model. The model

[4]https://github.com/niddal-imam/End-2-End-image-spam-detector-pixel$_l ink$

uses black and white lists that are updated whenever a new word is detected. As we focus on health-related spam campaigns, the blacklist initially contains the most frequent spam words used by these campaigns and the dataset described in Section 5.4. Also, the white-list initially was created using a Wikicorpus, which contains several Wikipedia articles. The corpus was tokenized and used as a white-list. However, these lists will be updated by a security analyst regularly.

5.4.3.0.4 Ethical issue. Images collected from Twitter were posted by users in trending hash-tags. Users account from which these images were collected have not been analysed in this research. Thus, this research is not involving Human Subjects. Also, some examples for images collected from Twitter were presented in this chapter to help the readers understanding the problem that this research is trying to solve.

5.4.4 Discussion

Recent studies have shown that when deploying DL-based models for cybersecurity systems, they become vulnerable to different adversarial attacks. Consequently, two security countermeasures were taken into account when designing the adversary-aware OCR-based detector of spam images. First, the robustness to the identified adversarial spam images was considered. The performance evaluation section shows that the robustness of the adopted OCR has improved by using adversarial spam images. Second, we consider how the detector can evolve over time in the face of emerging attacks (i.e., images with embedded modified spam words). Hence, we proposed an adaptable and interpertable text classification model to detect evolved threats and enable a security analyst to debug the detector when it is needed. The proposed text classification model consists of a blacklist/ whitelist with Human-in-the-loop to detect embedded modified or crafted words (adversarial spam images). Additionally, the detector is trained in a two-step fashion, where each part of the model is trained separately. This is different from end-to-end models that use a single training dataset to train the model. This training process will ensure that if part of the model is attacked, the other parts would not be affected.

Unlike most of existing spam image detectors, the developed adversary-aware OCR-based detector was designed considering the robustness to the identified adversarial spam images, the adaptability and human-interpertability to detect a new embedded manipulated text and ensure the detector can evolve over time. The detector can be used for detecting images with embedded malicious content in different social media platforms, such as Twitter and Facebook, which published a statement about detecting misleading health images [130]. The detector was not only designed for detecting spam images in Twitter; it was designed to extract text from any type of images and classify the extracted text. Also, it was designed to detect spam images with embedded English and Arabic text. Additionally, the developed OCR-based detector can be used for several purposes. For example, it can be used for training text recognition models. New or modified words can be used to update the CRNN and enable it to recognize new words. Also, it can be used as a defence method against causative attacks, in which an adversary contaminates training data to cause misclassification. It can be used to filter out contaminated samples that may be injected into the collected data for training ML models. Additionally, it can be used as either a Server-based or Client-based detector.

For example, users can define the list of words that they want the system to block images that contain any of the word in the list. Then, if a spam word is detected, the system will notify the user that this image is a spam image, or if a new word that cannot be found in neither the blacklist nor the whitelist is detected, the new word will be stored to be evaluated by the security analyst and the image will be muted and the user will be notified that it may contain sensitive content.

5.5 Summary

An approach for designing an adversary-aware OCR-based detector of spam images in Twitter through extracting and classifying text embedded in an image was presented. Text detection and recognition models that are commonly used in the literature were adopted to build our OCR system, and a new method for classifying extracted text from images was proposed. Experimental results show that we can improve the robustness of existing OCR systems by using adversarial spam images for fine-tuning (i.e., adversarial training). Also, since cybersecurity systems are adversarial environments, we designed our model to detect possible embedded text manipulation. To our knowledge, this the first work to design an adversary-aware OCR-based detector of spam images with Arabic embedded text uploaded on Twitter. More importantly, the developed detector was designed to be robust against adversarial spam images, adaptable and interpretable to detect evolving attacks.

In terms of limitations and future works, a couple of points need to be improved in the text detection model, such as detecting the maximum text size and detecting differently oriented text. Also, recognizing multiple language text is an area that needs to be improve in this model. One of the issues that has been found when building model for recognizing Arabic and English text is that the model mistakenly recognizes some Arabic letters that have shapes close to some English letters' shape. As the text classification model proposed in this chapter is based on blacklist, it may classify non-spam massages as spam (false positive) due to finding a spam word regardless of message's context. A possible solution for this drawback could be to examine another characteristic of a spam message along with the appearance of spam words in the image. For example, if a single spam word is detected in an image, account features (e.g., number of friends, or account reputation) and the message's content features (e.g., number of hashtags, or number of words) need to be checked. Another possible solution is to notify users by hiding the spam message and notify the user that the message may contain sensitive content, so the user can make the judgment. However, future work will be focused on improving the text classification part of the model.

Chapter 6

Spam Images Detection: An OCR Post-Correction for Improving the Robustness to Adversarial Text Images

In this chapter, the fourth contribution of the thesis is presented. After analysing the targeted spam campaigns in Twitter Arabic hashtags, we found that they manipulate embedded text in images. Based on this observation, we assume that adversaries may manipulate images embedded text by flipping some characters with visually similar symbols or number to fool spam image detectors. Hence, we proposed an OCR post-correction algorithm for improving the robustness of OCR-based detectors. Also, a text classification model was developed to ensure that the designed OCR-based detector is adaptability to evolving attacks and human-interpretability for debugging . This chapter includes and expands on our unpublished paper:

- Imam, N. and Vassilakis, V., Kolovos D., OCR post-correction for Detecting Adversarial Text Images, *Information Security and Applications*

6.1 Introduction

Extracting and understanding text in images (e.g., printed and handwritten documents, or natural scene text) is an area of study that has been widely researched in recent years, due to the increasing amounts of images shared on Online Social Networks platforms. OCR has been the leading technology to extract text embedded in images. The general structure of OCR systems consists of two components: text detection and text recognition. The success of OCRs in extracting text from cleaned documents has led to its adoption as a prepossessing step in many real-world applications, such as Neural Machine Translation (NMT) [151], license plate recognition [286], cancer classification [294], and recently, spam detection [49].

The process of classifying images with embedded text using OCR systems involves three steps: text localisation (detection), text recognition, and text classification. There are two main challenges of using OCR systems for image classification in an adversarial setting, choosing the best text extraction and text classification methods. First, extracting embedded text can be either at sentence-level or word-level. In the first method, an OCR reads the image line-by-line, and the detected text gets checked and corrected by a lexicon-based transcriptions [234], whereby the prediction is constraint to a spellchecking dictionary. As most of the text classification models depend on the input context when making their prediction, extracting text from images at sentence-level is suitable for scanned documents, where the embedded text is distributed in lines (see Figure 6.1). However, word-level text extraction, in which embedded text is extracted as a list of words, is more suitable for images found in OSNs. Figure 6.2 depicts some examples of images with embedded text that is distributed all over the image being posted on Twitter. Extracting the embedded text from such images by the OCRs that extract text at sentence-level is, however, not applicable. Thus, related studies in detecting spam image use a word-level OCR [48]. Nevertheless, classifying list of words is another challenge, as it could lead to higher false positive rates. For example, the simplest way of classifying a list of words is by using a blacklist, but spam words (e.g. Viagra or call) might appear in non-spam images. Advanced text classification methods, such as FastText, Byte-per Encoding (BPE), and Embedding have been adopted to classify extracted text from images. However, these Natural Language Processing (NLP) models have shown to be vulnerable to text perturbations [240].

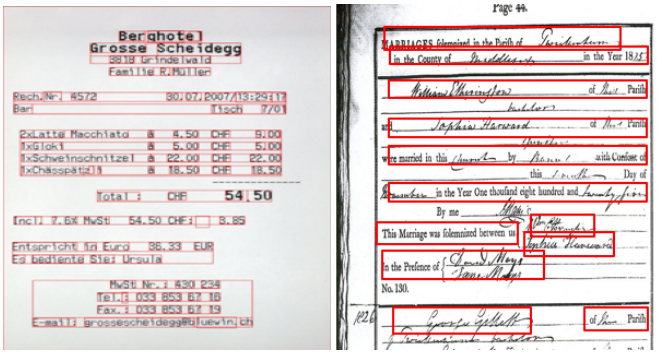


Figure 6.1: Examples of Scanned documents

Figure 6.2: Examples of Twitter's images

Another challenge is the robustness against adversarial examples. There is a rapidly growing concern about test-time attacks against neural network image classifiers. Some studies investigate black-box model queries type of test-time attacks, whereas others study the gradient-based optimization type of attacks [219]. These attacks can either be targeted (i.e., designed for a specific model) or non-targeted. There are two types of adversarial image examples that can be crafted against image processing models: adding perturbations to the images pixels, or manipulating the embedded text in images. Real-world applications are highly dependent on the correctness of the OCR outputs. Mistakes could lead to very serious consequences. For example, a parking fine ticket could be issued to a wrong car; incorrect medical diagnoses might be produced, or messages may be misclassified as spam [48, 27, 240]. Previous works focus on devising adversarial examples against OCR systems by adding perturbation or noise to images using Fast Gradient Sign Method (FGSM), DeepFoll, or Generative Adversarial Network (GANs). However, analysing extracted text from the OCR systems has not been well studied in the literature. This type of attack affect OCR-based systems at different levels.

OCR systems use either lexicon-based or lexicon-free methods for text recognition tasks. Lexicon-based OCRs use predefined lexicons and they could be useful for some tasks concerned with only producing outputs that are likely to be words in the target natural language; whereas, lexicon-free OCRs use Connectionist Temporal Classification (CTC) [199]. Lexicon-based OCRs may fail if used in cybersecurity systems, as they cannot recognise manipulated texts, and do not help security analysts to identify evolving attacks, as these systems are perceived as black-box. Although some lexicon-free OCRs can recognise manipulated texts, their output may fool the deployed text classifier. These text classifiers (e.g., NLP applications) are very sensitive to certain words when making predictions, so a small manipulation on the image's textual content could cause the OCR-based system to misclassify the image [282]. Song and Shmatikov [240] stated that there are no automated systems that could check whether the text produced by OCR "makes sense". Also, they mentioned that systems sensitive to out-of-context types of text manipulation, would be prone to false positives and adversarial attacks. Li et al. [171] added that a few defence methodologies have been proposed for adversarial text attacks. Kurita et al. [162] stated that adversarial text attacks are different from users' errors (e.g., misspellings and slang), as users do not initially attempt to avoid being detected.

While existing works investigate the effect of adversarial text attacks against NLP applications, it is more challenging to handle such attacks against OCRs. Related studies discuss two potential defence approaches, spelling check and adversarial training [171, 162]. In adversarial training, we apply noise or generate adversarial examples to the training dataset. One of the limitations of this method is the need for the defender to identify details of the incoming attack, such as the strategy and lexicon used by the adversary. Also, the model is trained on adversarial training fashion, which could be over-fitting to the adversarial examples; thus, leading to worse performance on clean datasets. On the other hand, using spell checking algorithms is the most common defence method against character-level perturbation in NLP tasks [171]. Although spelling check methods could detect and correct errors or adversarial examples, they cannot be applied in all domains, since their performance varies depending on the type of misspelling, such as deleting or substituting more than one character [13].

In this chapter, we investigate the effect of adversarial text images (i.e., images with manipulated embedded text) to OCR-based systems. This type of attack has been discussed in the literature against NLP applications using different names, such as character flips [127], scramble text [162], substitute characters [171], or visual perturbations [98]. We generate adversarial text images type of adversarial attack against OCR-based systems, where adversaries manipulate embedded text in an image by replacing a few characters to cause the text recognition part of the OCR (e.g., CTC) to misrecognise the text. We assumed that the adversaries have very limited knowledge about the deployed OCR (black-box setting). Components of automated OCR-based systems are rarely checked by humans, which makes them vulnerable to adversarial attacks. Adversaries can take advantage of this vulnerability to attack different parts of OCR-based systems [240]. These attacks are common in Online Social Networks (OSNs), as they do neither require any knowledge about the deployed model, nor any linguistic knowledge and human understanding [98]. In the previous chapter 5, we found that the health-related spam campaigns on Twitter Arabic hashtags purposefully misspelt spam words that are embedded in images (see Figure 6.3). As a countermeasure, we proposed a text classification step for OCR-based spam detector, in which we used a black/white list method with human assistance to detect new or modified words (adversarial examples) [141]. However, to reduce human intervention, a spellchecking-based algorithm was employed as an OCR post-correction step for denoising and classifying malicious text embedded in images was proposed. After the embedded text in an image gets extracted by the deployed OCR system, the proposed defensive algorithm denoises the manipulated (adversarial) embedded texts and classifies them before they feed into the text classifier model.

Figure 6.3: Images with embedded manipulated text

The study in this chapter aims to answer the following research questions: *RQ1*) Is the recognition accuracy of the SOTA OCR systems affected by the generated adversarial text images? *RQ2*) Are autocorrectors (i.e., spell-checkers) sufficient for OCR-based systems to mitigate the generated adversarial text images? *RQ3*) How can we improve the robustness of OCR-based systems to mitigate the generated adversarial text images? Our main contributions are as follows:

1. We developed a black-box attack method that can generate images of manipulated embedded text to cause OCR-based systems to misrecognise the text.

2. Human perception of the generated adversarial text images is an important feature of adversarial examples. We evaluated the perceptibility of our adversarial images by conducting a user study, and the results showed that human understanding is not affected by the manipulations.

3. We proposed an OCR post-correction algorithm for denoising and classifying malicious embedded text in images. Specifically, the proposed method has been designed to improve the robustness of OCR-based detectors.

4. We developed an adversary-aware OCR-based detector that is robust to adversarial text images, adaptable and interpretable to evolving attacks.

The rest of this chapter is organised as follows: Section 6.2 discusses related work in the field of OCR systems. In Section 6.3, the problem formulation is presented. The datasets, methodology, and our proposed algorithm are presented in Section 6.4. We quantitatively measured human perception of the generated attacks and present our findings in Section 6.5. Section 6.6 presents the results and analysis of three experiments preformed to answer the research questions. The discussion of the experiments' results and its limitations is presented in Section 6.7. Finally, Section 6.8 summaries the chapter.

6.2 Related Work

Related work to our study is divided into two folds: adversarial text attacks and defensive methods in images classification and text classification tasks. Since there is paucity of research that investigates adversarial text attacks against OCR-based systems, we also discuss similar attacks against NLP applications. OCR-based systems often use NLP applications as the last component of the system. Thus, it is important to understand attacks and defence methods proposed against these applications.

6.2.1 Adversarial Text Attacks and Defence in Images Classification Tasks

Adversarial text attacks against OCR systems could be carried out either by adding noise to the text locations in images, or by directly manipulating the embedded text in images. A large number of works have been studied in a bid to examine the former attack. However, to our knowledge, no previous study has been conducted towards investigating the later type of attack. Song and Shmatikov [240] presented the first study of adversarial examples against sequence labelling models in the image domain. They proposed several gradient-based adversarial attacks against CTC-based OCRs that uses Tesseract-ocr, by adding perturbation to the most influential characters or words in the scanned documents. The authors successfully caused Tesseract-ocr to output the desired adversarial texts with 84.8% accuracy. Also, they showed that their attack could affect the prediction accuracy of NLP applications that uses OCR systems for pre-possessing. One limitation of their attack is the lack of transferability to different OCR systems.

6.2.1.1 OCR-based Detectors

Some recent studies proposed multi-stage OCR systems for detecting images with malicious content in OSNs. OCRs extract text from images through: text detection and text recognition. In the text detection stage, designing features to distinguish text from backgrounds is the main task. The traditional methods design features manually to capture the embedded textual information. However, in deep learning methods, features are learned directly from training data [289]. The text

recognition stage extracts and predicts the textual information of the detected text through two steps: sequence modelling (e.g., LSTM) and prediction (e.g., CTC or attention-based (Attn)) [17].

Authors in [49] proposed one of the first studies that developed an OCR system for detecting and recognizing text in images uploaded to Facebook. The system, called *Rosetta*, consists of two models: text detection and text recognition. The text detection model uses Fast Convolutional Recurrent Neural Network (Fast-RCNN) to perform word detection. Then, for each detected box, a fully-convolutional model, referred to as Connectionist Temporal Classification (CTC), is used to recognize text [117]. The recognition model predicts the most likely character at each detected box in the image. Also, [282] developed a model called *Malena*, which can detect different types of spam including images carrying text, number, or QR code in Chinese social networks (Baidu, Tieba, and Sina Weibo). The authors used a PixelLink-based OCR [87] for detecting text in images. They generate 200 adversarial examples using the CW approach to evaluate the robustness of their OCR, and they successfully detected 196 of them. Tramer et al. [255] developed a framework for blocking adversarial ads in Facebook and web pages in general. The proposed framework takes a screenshot of the page, locates images by using off-the-shelf object detector, Yolov3; and then extracts text from images using Tesseract OCR. The authors evaluated the robustness of the framework against different evasion attacks. They evaluated the robustness of Tesseract OCR against the CW attack (ℓ_2 norm).

Although these studies employed OCRs for cybersecurity tasks; the vulnerability of these cybersecurity models to manipulated textual content of images has not been investigated. *Rosetta* was designed to detect spam images, but the vulnerability of the proposed OCR has not been evaluated. The authors of [282] and [255] evaluated the robustness of their OCRs using gradient-based attacks (e.g., adding noise or blur to images). To the best of our knowledge, our work is the first to investigate the robustness of OCR-based systems against images with manipulated embedded text.

6.2.1.2 Post-OCR Text Correction

There are several studies that used post-OCR correction to correct (denoising) OCR-ed texts. One of the post-OCR correction methods is to use spell checkers for correcting OCR's errors [266] [211]. Taghva and Stofsky [249] proposed spelling correction system (OCRSpell) for correcting OCR errors in text. The system uses Longest common subsequence calculation to correct errors of segmentation part of the OCR, such as iii $\rightarrow$ m , or cl $\rightarrow$ d . In [50], authors proposed post-processing steps to improve OCR's accuracy using the Aspell API and a customized words list. Thompson et al. [251] proposed a customised OCR correction for Historical Medical text. The authors compared the results of 4 spelling correctors: ASpell, Hunspell, Microsoft Word, and MAC OS. Additionally, a couple of competitions were organised by International Conference in Document Analysis and Recognition (ICDAR) on post-OCR text correction. Participants were asked to perform two tasks: OCR-error detection and correction. Several methods have been proposed in ICDAR-2019, such as context-based Character Correction using Bidirectioal Encoder Representations from Transformers (BERT) and dictionary-based detection.

The related studies use a post-correction algorithm to improve the recognition of OCR systems that are designed for analysis of the scanned documents. The tasks of extracting and understanding

text in scanned documents are easier than images uploaded into OSNs. Images uploaded into OSNs are more nosier as they could contain overlaid texts on top of images. Also, the embedded text in images cannot be read line-by-line, because the rotation of the embedded text can either be vertical or horizontal. Additionally, these studies showed that applying automatic spellchecking to misspelt words alone is unreliable. Some of the shortcomings of spellcheckers that need to be addressed include, complexity, out-of-vocabulary words (OOV), and many others. For example, OCRSpell is complex and requires extensive feature engineering, as its operation involves five models. Other studies [50, 251] have not considered detecting and tracking manipulated texts. In this study, we focused on OCR systems designed for analysing images with embedded text. The perturbation of multiple characters in a document does not affect human understanding, as they can infer the meaning of the perturbed words from the context. On the other hand, images with embedded text tend to contain less text, which is a constraint that needs to be considered by adversaries when adding perturbations to text in images. In contrast to related studies, we use an OCR-post correction for security purposes, whereas they use it for improving the models accuracy.

6.2.2 Adversarial Text Attacks and Defence in Text Classification Tasks

Character-level perturbation attacks against text classification models have been widely studied in the literature. These attacks can be launched as either in a white-box or black-box setting. Heigold et al. [127] proposed several character perturbation methods for attacking NLP models (e.g., bpe-LSTM-BLSTM, char-LSTM-BLSTM and char-CNN Highway-BLSTM). They found that character-based approaches are more sensitive than BPE-based approaches. Ebrahimi et al. [93] proposed the HotFlip method for generating adversarial examples against character-level neural classifiers. The attack targets the one-hot encoding step of the character-level embedding process. They chose the best character to be flipped by computing gradient with respect to one-hot encoding. Also, they used a beam-search optimisation to find a set of manipulation (flip, insert, delete) that could fool the deployed classifier. Moreover, the authors extended HotFlip in Ebrahimi et al. [92] by adding targeted attacks against character-level Neural Machine Translation (NMT). The authors concluded that adversarial training could improve the robustness of the deployed text classifier against such attacks. Similarly, Miyato et al. [197] investigated the robustness of Recurrent Neural Networks (RNNs) by perturbing the continuous word embedding, rather than the discrete word inputs, such as one-hot vectors. Their results showed that adversarial and virtual adversarial training improved the classification performance and the quality of word embeddings. Also, Gong et al. [115] proposed adversarial text attack against Convolutional Neural Network (CNN), with only a few words changed using FGSM and DeepFool. On the other hand, Belinkov and Bise [27] proposed several black-box attacks against NMT, such as swap, middle and fully random, and keyboard type. They treated typos and misspellings as adversarial attacks. Their experiments showed that models trained on mix noise performs worse than models trained on a specific type of noise.

Unlike previous works that used projected gradient, the authors in [113] proposed a novel framework, DeepWordBug, which generated character-level perturbation on text data. The authors swapped, substitute, delete, and insert characters, to generate Out-Of-Vocabulary (OOV), in a bid to force RNN to classify text as unknown. Their method of attack successfully reduced the

prediction accuracy of word-LSTM and char-LSTM models. Also, they concluded that adversarial training could improve the robustness of these models against their attack better that using auto-correctors. Li et al. [171] proposed adversarial attacks against Deep Learning Text Understanding (DLTU) in black-box and white-box settings. They proposed five bug generation methods: insert, delete, swap, substitute-character, and substitute-word. The authors evaluated their attacks against spelling checker and adversarial training. The results showed that spelling checker could be used against adversarial text attacks. Whereas, the effectiveness of adversarial training reduced in defending against unknown adversarial attacks. Schuster et al., [219] proposed a non gradient-based attack at training time, in which they changed words locations in the embeddings to misclassification. The authors investigated two defence methods: using anomaly detection in word frequencies and filtering out high-perplexity sentences. Although they found that filtering out sentence with ungrammatical sequence of words was the better defence method; whereas, using conjection-based and perplexity-based type of poisoning attacks could evade such detection method. Rojas-Galeano [214] proposed a method for detecting the homoglyph anomaly, which is a type of text manipulation intended to circumvent verbatim-based filters (e.g. sma!l or s.m.a.i.l). The authors used a penalty function crafted specifically to homoglyph substitutions-type of manipulation, aimed at detecting and tracing the locations of the potential obfuscations in text. The function takes users' generated text and an obscenity (vulgarity) as inputs and computes the edited distance between the two. The proposed edit penalty function compares between characters of the two inputs, and if admissible substitution symbols or bogus segmentation characters are found, these characters' positions are assigned 0, otherwise the positions will be 1. Although the proposed homoglyph/segmentation–safe distance (HS-dist) outperforms Levenshtein distance (L-dist) in discovering homoglyph similarities between obfuscations and obscenities, HS-dist runs four-times slower than L-dist.

These attacks used norm restrictions to ensure the validity of the perturbations. However, applying the same method for crafting adversarial text examples require some adaptations. A survey conducted by Zhang et al. [287] provides the difference between attacking Deep Neural Networks (DNNs) using adversarial images and text. There are two main challenges when generating adversarial text: 1) the gradient-based adversarial attack cannot be directly applied to the discrete data; 2) the level of manipulation is constrained by human perception [267]. The closest attack method to the attack investigated in this chapter is the DeepWordBug [113], but our attack targets OCRs and we focused on replacing characters with visually similar symbols or numbers. These studies have shown that adversarial training is very effective against these gradient-based attacks. However, such defence requires knowing details about the incoming attacks, and the trained model may become overfitted to the adversarial examples [171, 240]. Additionally, the possible forms of a manipulated word makes the training dataset more sparse [127]. The authors in [219] discussed defending adversarial text attacks by measuring the perplexity of sequences (i.e., how linguistically likely a sequence is), and they found that adversaries could evade such methods by deliberately reducing the perplexity. The algorithm proposed in [214] requires an extra step to find the matching words before calculating the edited distance. Also, it requires manually building a blacklist of potential words that could be manipulated by adversaries. Since calculating the edited distance takes time, adding extra step is not preferable. Hence, in this research we used spelling checker instead of adversarial training not only to improve the robustness, but also to ensure the model can evolve over time.

Table 6.1 summarised the related work to this research in terms of model used, type of attack and defence method.

Table 6.1: Related work that studied Adversarial text attack either against OCR-based or NLP-based Applications

Title	Model	Attack	Defence
Rosetta- Large Scale System for Text Detection and Recognition in Images [48]	OCR-based system for detecting spam in Facebook	None	None
Stealthy Porn- Understanding Real-World Adversarial Images for Illicit Online Promotion [282]	OCR-based framework for spam images in Weibo	C&W	None
AdVersarial- Perceptual Ad Blocking meets Adversarial Machine Learning [255]	OCR-based framework for detecting adversarial ads in facebook and the web page	C&W	None
Fooling OCR Systems with Adversarial Text Images [240]	OCR-based for license plate recognition system and scan document	optimization-based	None
Strategies for Reducing and Correcting OCR Errors [214]	OCR-post correcter model for OCR error correction	None	None
Improving OCR Accuracy for Classical Critical Editions [50]	OCR-post correcter model for OCR error correction	None	None
A Tool for Facilitating OCR Postediting in Historical Documents [249]	OCR-post correcter model for historical documents correction	None	None
On Obstructing Obscenity Obfuscation [214]	Verbatim-based filters	character level perturbations	Spelling correction-based
Towards Robust Toxic Content Classification [162]	NLP-based model for toxic comment detection	character level perturbations	Denoising Autoencoder
Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers [113]	NLP-based classifier	character level perturbations	None
TEXTBUGGER- Generating Adversarial Text Against Real-world Applications [171]	Deep Learning-based Text Understanding (DLTU)	character and word level perturbations	None
Our paper	OCR-based system for detecting images in Twitter	character level perturbations	Spelling correction-based

6.3 Attacks Against OCR-based Detectors

6.3.1 Problem Formulation

Given an input image with embedded text sequence $X = [X_1, X_2, ..., X_N, ...]$, where X is a sentence that consists of a number of tokens (i.e, words) X_N, and N is the number of words in a sentence. Each word X_N consists of characters $X_N = [x_1, x_2, ..., x_n]$, where n is the number of characters in the word. The deployed OCR model f scans the input X image's contents and outputs the class of the image $Y = [Y_1, Y_2, ..., Y_M]$, which is a predicted sequence of words, and M is the number of predicted tokens. $Y_M = [y_1, y_2, ..., y_m]$, where m is a predicted sequence of characters in a Y_M. The neural network of CTC that is used by the OCR outputs a sequence of probability vectors

for each word $f(X_N) = Y_M$ where $y_m \in [0,1]^{|\Gamma|}$ is the probability distribution over all characters (alphabets) Γ at position m. Since the length of the input sequence n and predicted sequences m are not generally equal $m \geq n$, it is hard to measure $p(Y|X)$ from $f(X)$. Thus, a *valid alignment* a of y is used to measure $p(Y|X)$. If sequence $a = [a_1, a_2, ..., a_i]$ and $a_i \in \Gamma \cup \{blank\}$ can be turned into y by removing blanks, which symbolized by $-$ and sequential duplicate characters, a is considered a valid alignment of y. For example, [b, b, -, u, y, y] is a valid alignment for [b, u, y] [240, 117]. To this end, there are two problems OCR-based systems can face at two stages under an adversarial text attack.

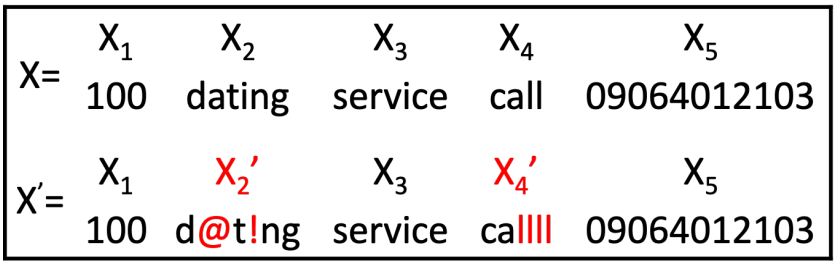


Figure 6.4: An example of adversarial text

At the text recognition stage, a pre-traind CTC-based OCR model f is used to map $X \rightarrow Y$. An adversary aims to launch an untargeted attack that causes the CTC neural network to misrecognize a few characters of the input sequence $x \epsilon X$ and predict invalid alignment $\acute{a}$ that is different from the ground truth $a \epsilon Y$. So that $f(\acute{x}) = a(a \neq y)$. For example, an adversary can manipulate one or two characters of an input image's textual content, which could cause the CTC to misrecognise the text since the conditional probability of CTC depends on the probability vectors and the ground truth label [199]. Figure 6.4 shows an example of character-level perturbation.

At the text classification step, NLP applications (e.g., BERT) are used to classify the extracted text. For example, in Figure 6.4 the manipulated embedded text of the input image $X = [X_1, X_2, X_3, X_4, X_5]$ is $X^{'} = [X_1, X_2^{'}, X_3, X_4^{'}, X_5]$. The manipulated words $X_2^{'}$ and $X_4^{'}$ could cause the deployed classifier to missclassify the input image. Even if we train the CTC to recognise manipulated characters, the extracted manipulated text could fool the deployed text classifier because NLP applications are sensitive to character-level perturbations. Utilizing adversarial training to improve the robustness of the deployed NLP application against this attack is challenging since the possible perturbed word formed make the training datasest more sparse. For example, flipping at most one character of a word of length n could make up to n^C different word forms, where C is the number of characters in the vocabulary [127]. Such an attack can easily be launched in OSNs as it does not require knowledge about the functionality of the deployed OCR-based system. The adversary only needs to know about the blacklist (e.g., sensitive words) used by the deployed model, which can easily be done in OSNs through *exploratory attack*, and then manipulate these words [142]. Hence, a pre-processing step that could check the extracted text from images before it being fed into the deployed text classifier is needed.

6.3.2 Threat Model

The adversaries' goal is to violate the integrity of the deployed OCR-based detector by obfuscating detection through manipulating images embedded text. The attack specificity can either be targeted or indiscriminate, but here we consider it targeted, in which the adversaries manipulate specific words (e.g., spam or toxic words). The adversaries knowledge about the deployed detector is assumed to be very limited (black-box attack setting). In OSNs adversaries could learn about the deployed detector by sending carefully crafted massages and use the detector's feedback to learn some of its characteristics [142]. However, they do not know the details of the deployed model. In terms of the adversaries capability, it is assumed that the adversary is only capable of influencing the deployed detector before the testing stage (*exploratory attack*). Although some studies showed that adversaries are capable of manipulating the training data (*causative attack*); we consider the more realistic attack, in which they can only influence the detector before the testing stage. This assumption can be generalised to different OSNs that utilize OCR-based detectors since these platforms share a lot of similarities. Generally, the components of OCR-based systems are easy for adversaries to reconstruct [240].

6.4 Our Method

This section describes the methodology used in this research. First, the datasets used for building and evaluating the proposed adversary-aware OCR-based detector. Then, we present the algorithm used for generating the adversarial examples. Finally, the proposed defence method is described.

6.4.1 Datasets

We study adversarial examples of images with embedded text on three benchmark datasets created for text classification tasks. The testing datasets were partially manipulated and used for evaluation. These datasets were chosen because they are widely used in related studies that focused on detecting malicious activities (e.g., spam, toxic, and offensive comments detection).

SMS Spam Dataset [1]: We used the dataset built by Almeida et al. [11], for building our images dataset. The adopted dataset is a collection of SMS massages collected from the Grumbletext Web site, which is a UK forum created for cell phone users to make public claims about SMS spam messages. The dataset consists of 5,574 short messages; 4,827 ham and 747 mobile spam messages.

Jigsaw[2] dataset: It consists of 215,000 annotated comments from Wikipedia, and it was used in one of Kaggle's challenges (Toxic comments classification Challenge). The dataset contains six attributes (toxic, severe toxic, obscenity, threat, insult, and identity hate). However, we only use one of them which is a general toxicity label.

The offensEval 2019[3] dataset: It consists of 13,240 tweets that have been annotated through crowd-sourcing. The dataset is labelled into two classes: offensive and non offensive tweets [162]. It was built for identifying and categorising offensive language on Social Media.

[1] https://www.kaggle.com/uciml/sms-spam-collection-dataset/data
[2] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
[3] https://competitions.codalab.org/competitions/20011

These datasets were used to create our image datasets since there are no publicly available datasets that suit our needs. Examples of some of the generated images are presented in Figure 6.5. All the generated datasets were published at Mendeley Data [137]. The following section discusses the steps preformed for building our dataset.

Figure 6.5: Images with Manipulated Embedded Text

6.4.2 Adversarial Text Images

Defining gradients in symbolic text is hard [113], so we generated adversarial examples (i.e., images with embedded manipulated text) directly on the images content through the following steps. First, we used Algorithm 1 for automatically generating adversarial text. The algorithm scans the input's characters, and if a character in the input matches one of the characters in the mapping list, it flips the character. The list includes the following: !, @, $, 1, 0, and more symbols can be added. If an input word includes any of the characters in the character map list, that character will be flipped to its visually similar symbol or number. We chose the most frequent words in SMS Spam dataset, toxic words in Jigsaw dataset, and offensive words in OffensEval 2019 dataset using Term Frequency-Inverse Document Frequency (TF-IDF). Then, we used a synthetic data generator [4] for embedding the manipulated text into images. Figure 6.5 provides examples of the adversarial examples used in this study

Algorithm 1: Adversarial Example Generator

Input: word $X = (x_1, ..., x_n)$
Output: adversarial word $X^{'}$

```
/* most frequent pairs of letters mapping dictionary                    */
SymbolMap_Dict = [a: @, s: $, i: !, l: 1, o: 0]
for each char x_n in word X do
    if char in SymbolMap. Keys then
        word.replace(char, SymbolMap.get(char)
end for
Return X'
```

6.4.3 Proposed Defence Method

Directly applying an automatic spellchecker to misspelled words is often unreliable. The suggested candidate words of the deployed spellchecker could be wrong or, the word list used in the spellchecker

[4] https://github.com/Belval/TextRecognitionDataGenerator

might not include some words. These limitations could correct the input word with an error [50], which could increase the false negative rate in security applications. Also, spellchecker cannot differentiate between typos or misspelling and text perturbations. Spellcheckers often correct mistakes and compute the edit distance, which could tell whether the mistake is addition or deletion. In our case, we want to know whether the misspelling/ error is an adversarial example (e.g., flipping, or swapping), or not and to keep tracking these adversarial examples so we can update our system. Another important point beside the robustness to adversarial examples that needs to be considered when designing a detection system in an adversarial setting is the ability to evolve over time in the face of emerging threats [230]. Thus, the spellchecker must either find the closest correction or print out the misspelling/ error, as it might be a new kind of text manipulation.

6.4.3.1 Proposed Algorithm

In order to overcome the above shortcomings, we propose an algorithm for improving the robustness to adversarial text images. The proposed algorithm adopted a basic spelling correction algorithm proposed by Peter Norvig [207]. Our algorithm denoises OCR systems' errors and outputs the class of the error detected in the text (e.g., repeated characters, swap characters, substituted characters, or OOV). Detecting the type of error helps in distinguishing between adversarial attacks, typos, or misspellings along with other adversarial activities that need to be considered, such as the importance of the word being manipulated. The process of the proposed algorithm involves three steps: lookup, scanning, and correction. First, it checks if the extracted word includes errors by looking up for a matching word in the dictionary. Second, if a matching word does not exist, the algorithm will check if it contains any symbols, repeated or swap characters to be corrected. Also, this scanning step tracks the type of error and checks if the corrected word exists in the dictionary. For example, if a symbol or repeated character is detected, the algorithm will substitute the symbol or delete the repeated character and correct/ denoise the text by using the adopted spell checker. The adopted spell checker finds the correction c, out of all possible candidate corrections, that maximizes the probability that c is the intended correction, given the original word w [207]:

$$argmax_c \ \in_{candidates} \ P(c)\ P(w|c)\ /\ P(w) \tag{6.1}$$

Finally, if the extracted word (i.e., input) cannot be corrected, it will be classified as OOV. As mentioned above, adaptability and interpretability of the detection system is important in an adversarial setting. Hence, extracted text that could not be corrected will be collected and be used by a security analyst for updating the system as these OOV might be new adversarial examples. The output of the algorithm will be the input correction and the class of detected error.

6.5 Human Perception

We quantitatively measured human perception of the generated images with embedded manipulated text. This is an important step when devising adversarial examples against ML-based models, as it ensures that the manipulations do not affect human understanding [171, 286, 264]. We conducted a survey, which consists of ten images (i.e., adversarial examples) and two multiple choice questions for

each image. We manipulated one character for the first five images and two characters for the rest. To avoid any bias, participants are asked whether they can understand the text in an image and only if their answer is YES, then they can choose the correct word. The total number of questions is 35. The survey's link was distributed using Facebook, Twitter, and Amazon Mturk

Survey Results. The total number of participants was 220, and participants of age 25-34 were the majority, with 46.33%. 55% of the participants were male and 45% female. 85.77% hold a Bachelor or higher degree. We first showed the participants the image and asked them if they can understand the text. Then, if the answer is YES, we asked them to choose the correct word. Participants were able to recognise the text with at least 91% accuracy, which means users' understanding is not affected by our adversarial examples. Also, 56.07% of them have encountered similar images, and 37.11% reported that such images could be found on social media. Moreover, 51.40% of the participants stated that they have used scrambled text when they are writing text.

The results of the survey showed that humans can recognise the embedded manipulated text in images. Also, these kind of images are found in social media more than in e-mails or SMS. Additionally, most of the participants have seen such images and used scrambled text.

6.6 Experimental Results and Analysis

We ran the experiments on Linux Ubuntu 18.04 LTS operating system with Intel(R) Core(TM) i7-8750H CPU 2.20GHz x 12 of 983.4 GB memory. Through experiments, our goals were to answer the three research questions that were defined in Section 6.1.

To answer the research questions, three different experiments were performed. First, we evaluate the robustness of our algorithm and five SOTA spellcheckers to four adversarial text examples and compare the results. Second, we launched our black-box attack against four SOTA OCRs. Third, we investigated using our proposed algorithm in OCR-based systems designed for detecting images with malicious contents (i.e., spam, toxic, and offensive).

6.6.1 Effect of Adversarial Text on Auto-correction

Since we chose to use a spellchecker in our defense method, a natural question arise: can spell checkers detect adversarial text? For this experiment, we used five spellchecking tools to correct four types of text manipulation methods. The tools are listed below:

1. **Peter Norvig** [207]: a spellchecking algorithm that generates candidates within 2 edited distance from the original word, and the highest frequent word is chosen as the correct word.

2. **SymSpell** [5]: a spellchecking tool that uses the Symmetric Delete spelling correction algorithm to reduce the complexity of calculating edit distances and looking up dictionary.

3. **Hunspell** [6]: one of the most popular spellcheckers used by LibreOffice, OpenOffice.org, Mozilla Firefox 3 Thunderbird, and Google Chrome.

[5] https://github.com/wolfgarbe/SymSpell
[6] https://github.com/hunspell/hunspell

4. **Pyspellchecker** [7]: a spellchecker based on Peter Norving's algorithm that uses a Levenshtein Distance algorithm and a list of frequency words to find corrections.

5. **Textblob** [8]: a python library provides different NLP tasks including spell correction, which uses Peter Norving's algorithm.

Original	Flipping	Swapping	Deletion	Insertion
Busy	Bu$y	Bsuy	Bsy	B*usy
Caller	C@1ler	Claler	Cller	C*aller
Reply	Rep1y	Rpely	Rply	R*eply
winner	w!nner	wniner	wnner	w*inner

Table 6.2: Some of the adversarial text examples used in experiments

Following the methodology used in [13] to test the spellcheckers, we generated four types of adversarial text including our attack using a list of top 20 frequent words in the SMS dataset. We used the DeepWordBug method proposed by Gao et al. [113] to generate three adversarial texts: (1) insertion: inserted one random character to the words (e.g., c*all), (2) deletion: we removed the second character (one was removed per word), and (3) swapping: we swapped the second and third characters in the word (one swap per word). Also, we used our developed method to generate the (4) flipping: we substituted one or two characters with visually similar symbols or numbers. Table 6.2 presents some of the adversarial text examples used in the experiments. The metric used for evaluating the performance of the spellcheckers is correction accuracy. Figures 6.6, 6.7, 6.8, 6.9 present the results of the six spellcheckers to correct/ denoise the four types of adversarial text. The experiments showed that the six spell checkers preformed the worst when we deleted a character from the words, whereas they preformed the best against the swapping type of adversarial text. The proposed algorithm outperforms the five spellchecking tools in correcting two types of adversarial text attacks: flipping and swapping. Also, it achieved the second best results in correcting deletion and insertion type of adversarial text attacks. Figure 6.6 shows that Pyspellchecker achieves the best result as it corrects 80% of the adversarial text examples (i.e., flipping), whereas Hunspell is the worst tool among the five spellchecking tools. It outperformed the five spellcheckers in correcting the targeted type of adversarial text attack (i.e., flipping) with 20%. Hence, in this experiment, we answered (*RQ1*) as the results showed that using an auto-correction tool against adversarial attacks is not sufficient. This findings support results of related studies [162, 113], and the results of the experiments show our modified spell checker outperforms the five spell checkers.

6.6.2 Effect of Adversarial Text Images on OCRs

In the second part of experiments, we demonstrated the effect of the generated adversarial text images against the text recognition part of OCR systems. Three benchmark OCRs were used to evaluate the effectiveness of the attack. In the text recognition part of the OCR, two techniques are commonly used for the prediction stage: CTC and Attention-based sequence prediction (Attn). These two techniques have been used in many OCR systems. Several related works used CTC-based

[7] https://pypi.org/project/pyspellchecker/
[8] https://github.com/sloria/TextBlob

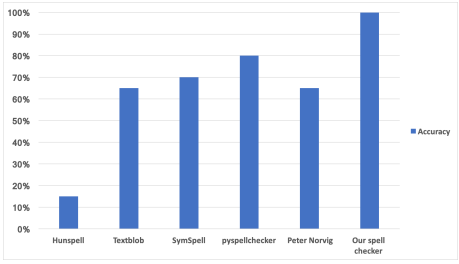


Figure 6.6: Flipping

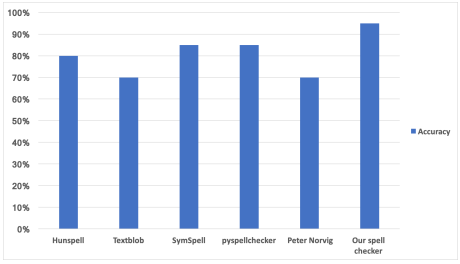


Figure 6.7: Swapping

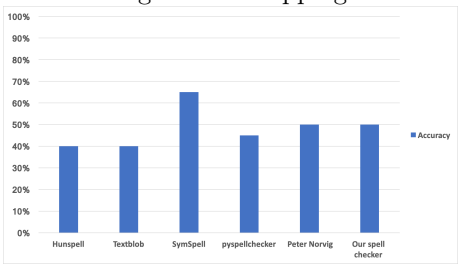


Figure 6.8: Deletion

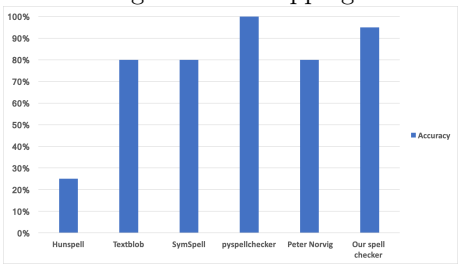


Figure 6.9: Insertion

OCRs for spam image detection, such as *Rosetta* [48] and Ads blocking [255]. To evaluate OCRs that use CTC, we chose *Rosetta* OCR, as it was designed for detecting spam images uploaded to Facebook. Also, *Tesseract* was chosen because it is an open source publicly available, and it is widely used in many OCR-based systems [255]. For Attn-based OCR experiment, we used Thin-plate-spline (TPS) based Spatial transformer network (STN)[17]. The TPS achieves the 1st place in ICDAR2013 focused scene text and ICDAR2019 ArT, and 3rd place in ICDAR2017 COCO-Text and ICDAR2019 ReCTS (task1). Two different versions of TPS were used: TPS-NS (non case-sensitive) and TPS-S (case-sensitive). Table 6.3 shows some examples of images with manipulated embedded text that are miss-recognised by TPS.

Table 6.3: Examples of Images with Manipulated embedded Text Misrecognized by Attn-based OCR

Adversarial Examples	**Prediction**	**Confidence**
1!ve	ilve	0.65
c@$h	cosh	0.86
C@1ler	coiler	0.85
1@test	10test	0.68

In this section, we evaluate the text recognition accuracy of the four SOTA OCR systems: *TPS-NS*, *TPS-S*, *Tesseract*, and *Rosetta*. First, we evaluate the recognition accuracy when using images with clean embedded text. Then, we evaluate the OCR systems when flipping one or two characters. Measurement metrics used for our experiments were Correctly Recognized Word (CRW) and Total Edited Distance (TED) [147]. CRW is the total number of correctly recognized words by an OCR system, whereas TED is a weighted sum of the Levenshtein distances between the correction of the OCR and the corresponding token in the Ground Truth [213]. The lower the total edited distance, the better. The results of our attacks against the four OCRs are presented in Tables 6.4 and 6.5.

Table 6.4 presents the results of the four OCRs without using the proposed OCR post-correction algorithm. The results showed that the recognition accuracy of the OCRs dropped significantly when manipulating two characters. These results proved that adversaries can launch a black-box adversarial text attack against OCRs without knowledge about their functionality and parameters (answer to *RQ2*). The highest recognition accuracy when manipulating two characters was achieved by TPS-NS, while the worst was achieved by Tesseract.

Table 6.4: Results of the Adversarial Images with Perturbed Text Against the three OCRs.The best results are highlighted in **bold**

Models	Perturbed Character	CRW	TED
TPS-NS	0	85%	6%
	1	**72**%	12%
	2	**54**%	28%
TPS-S	0	100%	6%
	1	51%	19%
	2	26%	46%
Tesseract	0	100%	100%
	1	21%	33%
	2	15%	60%
Rosetta	0	87%	5%
	1	72%	12%
	2	46%	20%

In Table 6.5, we evaluated the four OCRs using our proposed OCR post-correction algorithm. The results demonstrated that our algorithm improves the recognition accuracy of the OCRs by at least 10%. The best text recognition results achieved by TPS-S with 82% and 72% CRW. The results of this OCR system improves by at least 30% (from 51% to 82%) when manipulating one character of an input image, and 45% (from 26% to 72%) when manipulating two characters (answer to *RQ3*).

Table 6.5: Results of the Three OCRs after using the Proposed Post-correction. The best results are highlighted in **bold**

Models	Perturbed Character	CRW	TED
TPS-NS	1	82%	10%
	2	69%	19%
TPS-S	1	**82**%	9%
	2	**72**%	10%
Tesseract	1	72%	17%
	2	69%	25%
Rosetta	1	82%	10%
	2	69%	31%

6.6.3 An Adversary-aware OCR-based Detector

In the last part of the experiments, we investigate using a Multiple Classifiers System (MCS) to design an adversary-aware OCR-based detector that is robust, adaptable and interpretable. As

the generated attack can affect the text classification part of OCR-based detector, we evaluate the developed adversary-aware OCR-based detector using an MCS for the text classification task. Additionally, we design this part of the detector to ensure the adaptability and interpertability. For aggregating the output of the MCS, there are different methods that could be used, such as linear, non-linear, statistical, and ML combination methods [20]. Kurita et al. [162] used a linear aggregation method that makes its final prediction based on arithmetic mean of two models' predicted probabilities. Their results showed that the ensembled model outperforms a single classifier when tested on dataset that includes adversarial text. In ML aggregation method, a learner algorithm (e.g., DT, K-NN) that learns from the base classifiers' accuracy is applied as a higher level classifier. [272], discussed the potentials of using Multi Kernel Learning (MKL) for combining the output of multiple classifiers, visual and textual. The authors discussed the advantages and disadvantages of aggregating different deep learning based models using kernel learning. Voting rules, which is a non-linear aggregation method is widely used, and there are several combination rules, such as majority voting, weighted voting, minimum probability, maximum probability, multiplication of probabilities, and average of probabilities [188]. Using such methods adds more complexity to the detector, which makes it non-interpretable and default to be adaptable. Hence, we used a Fuzzy Rule Based (FRB) classifier for aggregating the outputs of the three classifiers similar to previous chapters.

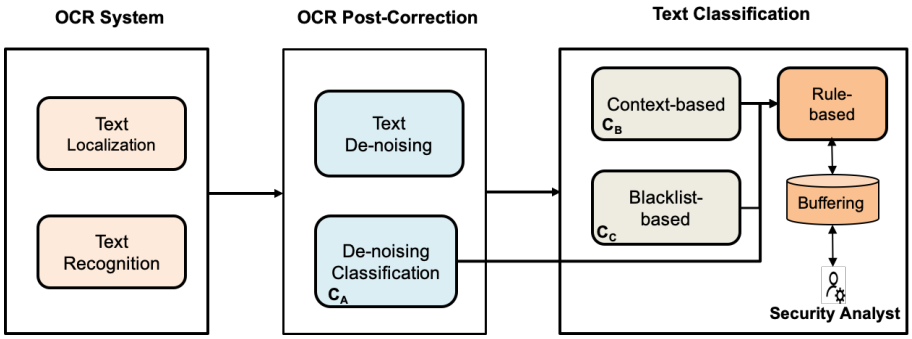


Figure 6.10: The Structure of the developed OCR-based Detectors

The developed adversary-aware OCR-based detector designed to extract text from images and to classify the images based on the embedded text. First, we discussed the components of the OCR-based detector and then we used it for toxic comments and offensive tweets detection. Figure 6.10 shows the structure of the developed OCR-based detector. It consists of three steps: OCR system, OCR post-correction, and text classification. In the OCR system, we use off-the-shelf tools, PixelLink [87], for localizing/ detecting text in images. It is a scene text detector model that has been used in [282]. For the text recognition, we chose the TPS-S (case sensitive) model since it achieves the best results in the above experiments. In the second step of our detector, the proposed algorithm was used for denoising and classifying errors or adversarial text. The last step is text classification, in which we used two text classifiers: context-based and blacklist-based classifiers. We used MCS for three reasons: (1) the output of the OCR system is a list of words, which can affect the accuracy of a context-based classifier, (2) a related study [162] shows that ensemble model outperforms a model that uses a single classifier, and (3) to ensure that adversary-aware OCR-based detector can evolve

over time in the face of potential new text manipulation attack.

The context-based classifier relies on the input words order when making its final decision and the blacklist-based classifier, which is a unigram model based on single words, and it is very sensitive to most frequent words. Consequently, we designed our detector to make its final decision based on the output of three classifiers: The results of our proposed algorithm, the context-based classifier and the blacklist-based classifier, which are fed into the FRB part of the text classification step for the final decision. The classifiers were trained in parallel using the same training dataset. The FRB will make its final decision based on the majority voting rules. For example, if one of the text classifiers and the proposed algorithm classification agree on an input X class Y, or if both disagree with the output of the proposed algorithm, the output of the two classifiers will be used for the final decision. Samples that the classifiers disagreed on will be collected and used by a security analyst to update the classifiers. Table 6.6 presents all possible outcomes of the classifiers and the associated set of rules. The denoising classification part of the OCR-post correction output four type of errors (swapping, flipping, repetition, or OOV). In this study, we consider text contains a combination of letters and numbers or letters and symbols (i.e., flipping), or a text contains a letter repeated more than two (i.e., repetition) as malicious classes. Thus, the output of the denoising classification part of the OCR-post correction will be 1 if a flipping or repetition is detected and 0 otherwise.

Table 6.6: Fuzzy Rules

Rule	Antecedent	Consequence
1	IF (C_A is 0) and (C_B is 0) and (C_C is 0)	THEN (Y is 0)
3	IF (C_A is 0) and (C_B is 1) and (C_C is 0)	THEN (Y is 0)
4	IF (C_A is 1) and (C_B is 0) and (C_C is 0)	THEN (Y is 0)
2	IF (C_A is 0) and (C_B is 0) and (C_C is 1)	THEN (Y is 0)
7	IF (C_A is 1) and (C_B is 1) and (C_C is 0)	THEN (Y is 1)
5	IF (C_A is 0) and (C_B is 1) and (C_C is 1)	THEN (Y is 1)
6	IF (C_A is 1) and (C_B is 0) and (C_C is 1)	THEN (Y is 1)
8	IF (C_A is 1) and (C_B is 1) and (C_C is 1)	THEN (Y is 1)

Where:

C_A: Output two classes (non-malicious: 0 or malicious: 1).

C_B: Output two classes (non-malicious: 0 or malicious: 1).

C_C: Output four classes (Swapping: 0, Flipping: 1, Repetition: 2, or OOV: 3).

X: An input (image).

Y: Class (0 or 1).

For generating adversarial text images, we used Algorithm 1. We selected 60 samples from each testing datasets (e.g., 40 non-toxic and 20 toxic), and manipulated the most frequent words. We only manipulated the toxic and offensive samples of the testing datasets since our threat model assumes that adversaries would only manipulate malicious samples.

After describing the components of our detector, we discuss the functionality. First, the OCR system part localises the text in an input image and extracts the detected text. Second, our proposed OCR post-correction corrects (denoises) and classifies the extracted text. The output of the text denoising part of the OCR post-correction is fed into the text classifiers. On the other hand, the

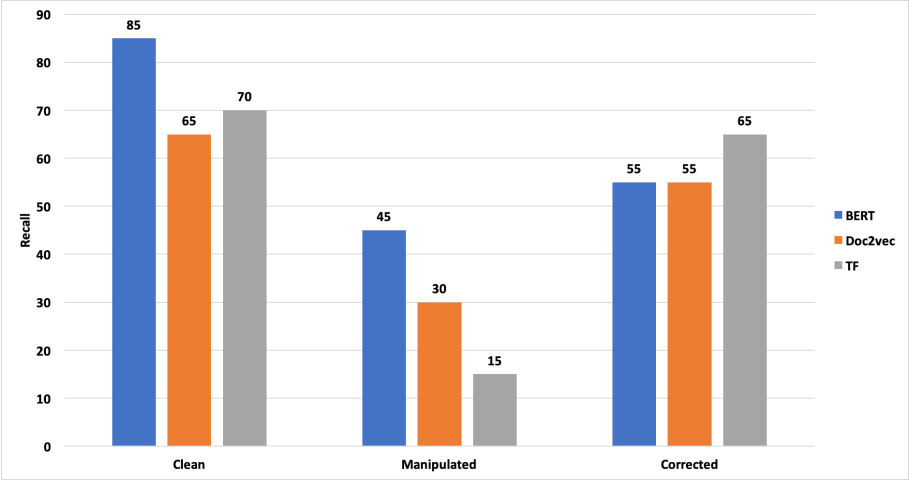


Figure 6.11: The results of the developed OCR-based detector using three different text classifiers that were evaluated on clean, manipulated, and corrected test dataset

output of the denoising classification part of the OCR-post correction, which is a type of errors (swapping, flipping, repetition, or OOV) is used along with the output of the text classifiers for the final decision. In the last step, we used the two text classifiers (context-based and blacklist-based), and the FRB for the final decision. In case of a disagreement between classifiers, the majority voting will be consider and the samples will be collected for updating the classifiers by the security analyst. In Figure 6.11, we compared the results of our OCR-based detector using three SOTA text classifiers. We used BERT, which has been used in related studies and shown to have the capability to handle OOV [162]. Also, several studies showed that Doc2vec [167], which captures the meaning within embedding, could detect spam tweets with high accuracy [258]. The third classifier was a simple logistic regression with unigram (LR). All classifiers were trained on clean datasets (i.e., do not include adversarial text) and evaluated on clean, manipulated and denoised test datasets. Following the evaluation method in [162], the results showed that BERT achieves the best results among the three classifiers. Also, the results showed that when using our algorithm for denoising the output of the OCR, the recall (i.e., ability of the classifier in detecting malicious samples) of the developed detector improved by at least 10%.

Original	OCR output
well go to hell	well hell to go
hahaha just shut up and learn how to read.	hahaha learn read how shut just and to up
I hate this class. This sucks	Ihate sucks class this This
what the hell are you doing	doing what you the are hell

Table 6.7: Examples of the original test dataset samples and the OCR's output samples

Also, we investigated whether the accuracy of the text classifiers used in the developed adversary-aware OCR-based detector gets affected by the output of the OCR, which is a set of words instead of a sentence. As we discussed earlier in Section 6.1, one of the adversaries techniques to fool OCR-

based detectors is to spread the embedded text all over the image (see Figure 6.1). Thus, to mitigate such adversarial activities, extracting a list of embedded words from images could be used. However, since we use a text classifier as part of the OCR-based detector, we compared the results of the three text classifiers using the original version of jigsaw and OffensEval-2019 test datasets and the OCR's output version of the test datasets. Table 6.7 presents some examples of samples from the jigsaw test dataset. It shows the original sentences and the output of the OCR. Figures 6.12 and 6.13 compared the results of the developed OCR-based detector using the three text classifiers. The comparison was preformed using the two test datasets jigsaw and OffensEval-2019. After training the three text classifiers, we evaluated the classifiers on the original test datasets and the output of the OCR system. The results in the two figures showed that the performance of the context-based classifiers (i.e., BERT and Doc2vec) was affected by the change in the words order, whereas it is more stable when using the LR with unigrams. As the LR classifier is sensitive to most frequent words, we used MCS (BERT and LR) for the text classification task. In the last experiment, we compared the results of the developed adversary-aware OCR-based detector using a single classifier and MCS.

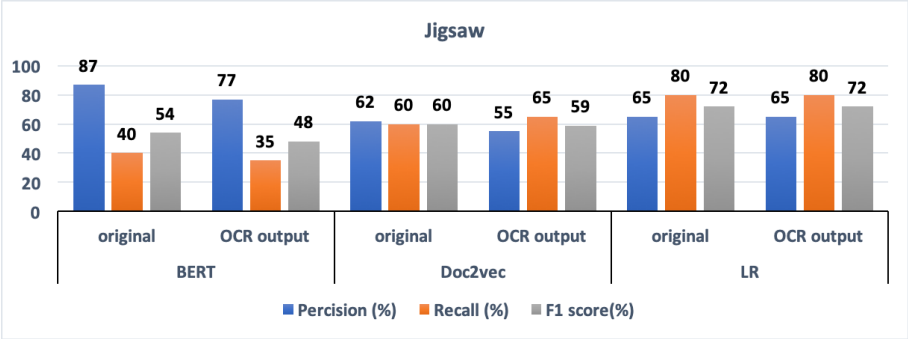


Figure 6.12: Effect of change in the words order on the text classifiers using the Jigsaw dataset

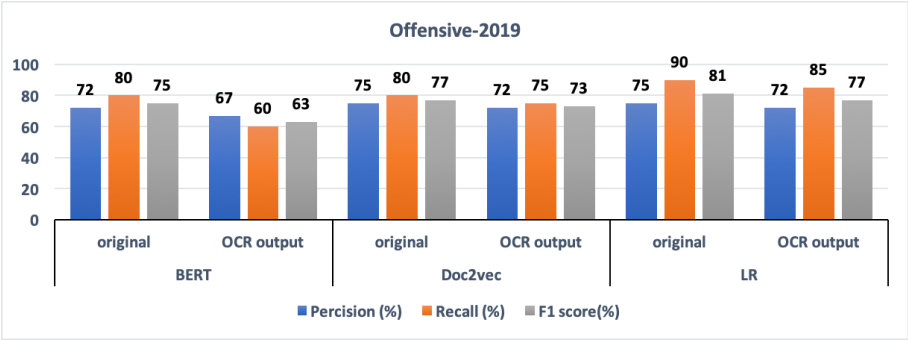


Figure 6.13: Effect of change in the words order on the text classifiers using the OffensEval-2019 dataset

In Table 6.8, we make the following comparison: the results of using a single text classifier and MCS with or without our OCR post-correction. We found that the developed OCR-based detector achieves better results when using MCS than using a single text classifier, in terms of Precision,

Recall, and F1-score. Also, each individual classifier results improved when using the OCR post-correction. In detail, the results in Table 6.8 showed that the performance (i.e., recall and F1-score) of three OCR-based detector using two datasets decreased under attack by at least 10%, while they improved by at least 10% when using our algorithm. For example, the recalls of Doc2vec under attack improve when using our algorithm from 30% to 55% and from 35% to 75% on Jigsaw and Offensive-2019 datasets respectively. We considered the recall and F1-score as our thread model assumes that the adversaries would only manipulated the malicious samples. These results reflected that the robustness of the OCR-based detectors have improved when our algorithm is used. Robustness here refers to the action of decision makers (i.e., text classifiers) through correcting/ denoising the output of the OCR system, and classifying the type of error.

Also, one of the goals of using MCS is to reduce the false positive rate of the detector. We can see in Table 6.8 that the precision of blacklist-based classifier (the LR) is lower than precision of context-based classifiers (BERT), because it is sensitive to most frequent words. Thus, to overcome this issue, we used the FRB that considers the majority voting of the two classifiers and the proposed algorithm classification of the corrected error. The results showed that the precision of the MCS is better than both classifiers' precision. Additionally, we conducted our investigation using the OCR-based detector with MCS to deal with evolving attacks and to be interpretable for debugging. The results in Table 6.8 show that using the MCS of BERT and LR with the proposed algorithm achieved better overall performance. The proposed OCR post-correction is designed to denoise errors and to classify the type of errors. When the error is classified as OOV by the proposed algorithm, but became classified as non-malicious by any of the text classifiers, this type of error will be used as an indicator of a new possible attack. Hence, these examples will be collected and checked by the security analyst to confirm if the detector needs to be updated or not.

Table 6.8: Detailed results of the developed adversary-aware OCR-based detector on two datasets and three test datasets: clean, manipulated, and denoised datasets. The best results are highlighted in **bold**. The MCS (BERT and LR) out-preforms using a single text classifier

Model	Testing Data	Jigsaw			Offensive-2019		
		Percision	Recall	F1	Percision	Recall	F1
BERT	clean	77	35	48	67	60	63
Doc2vec		55	65	59	72	75	73
LR		65	80	72	72	85	77
BERT	manipulated	77	25	38	67	15	24
Doc2vec		55	30	39	72	35	47
LR		65	15	24	72	20	31
BERT	Denoised	77	55	64	67	80	83
Doc2vec		55	55	55	72	75	73
LR		65	75	51	72	85	78
MCS		**87**	**85**	**86**	**82**	85	**83**

6.7 Discussion

The experiments preformed in this study showed how the proposed algorithm can improve the robustness of OCR-based systems to images with embedded manipulated text. Denoising the output

of the OCR systems could be helpful in the classification task. Most importantly, classifying the type of errors helps detecting adversarial examples. Although we focused on a particular adversarial attack, where adversary replaces characters with their similar looking symbols or numbers; our algorithm could be modified to detect different adversarial attacks. For example, it could be modified to detect an adversarial text attack, in which adversaries replace some characters in important words with asterisks (e.g., st**id). This has been shown in the first experiment (see Section 6.6). Also, the experiments show that the proposed algorithm can be used for improving the robustness of NLP applications. The results in the first subsection 6.6.1 of the experiments show that our algorithm outperforms five spelling checkers.

Generally, when designing an automated ML-based model in an adversarial environment, very important points that need to be taken into account at the design stage are the adaptability to evolving attacks and interpertability to security analysts. In other words, we need our model to evolve over time in the face of new attacks. The proposed OCR post-correction has been designed to detect OOV, which could be used as an indicator of a new attack. For instance, an increased number of OOV in images with non-malicious requires debugging the model to investigate if there is a new attack. Additionally, in this chapter, we show that the proposed algorithm can be used with any OCR-based detector designed for adversarial environments. For example, we investigate improving the robustness of Facebook spam image detector *Roseeta* using our OCR post-correction algorithm.

There are some limitations that future works need to consider. In order for our OCR post-correction to be effective, the OCR system needs to accurately recognise the embedded text in images. The recognition accuracy of OCRs gets affected by noise and adversarial perturbations. This is especially true for images uploaded into OSNs, which are images with text overlaid on top of them; extracting text from such images with high accuracy is an area that requires more research. Also, the robustness of the OCR-based systems against adversarial image attacks (e.g., FGSM or Adversarial Watermarks) needs to be considered. Additionally, one of the drawbacks of OCR systems when used for detecting spam images is that the embedded text gets extracted as a list of words, which affects the text classification. In this study, we used an MCS with the FRB classifier to tackle this issue. However, more research to find a method for re-ordering the extracted words need to be considered.

6.8 Summary

Using an OCR in cybersecurity systems is an important and challenging problem. In this chapter, we investigated the robustness of OCR-based systems against images with embedded adversarial text (i.e., adversarial text images). Similar to other domains application (e.g., NLP), we found that SOTA OCRs are vulnerable to such slightly manipulated embedded text that do not affect human's understanding. We described our proposed OCR post-correction, which denoises and classifies various types of errors to improve the robustness of OCR-based systems against character-level adversarial attacks. Our algorithm improves the text recognition of SOTA OCRs with over 10%. Additionally, as a case study, we showed how our algorithm could improve the overall performance of OCR-based systems designed to detect images with embedded malicious content (e.g., spam,

toxic or offensive comments). The developed adversary-aware OCR-based detector consists of an OCR and an MCS text classification model along with the proposed algorithm. As demonstrated in our experiments, our algorithm improves the accuracy of the detector by over 20%. From security point of view, utilising an OCR post-correction do not only provide robustness against adversarial examples, but it also provides scalability, adaptability and interpretability, which helps detecting unknown attacks.

The investigation of OCR-based detectors in this chapter reveals that there are some limitations that future works need to consider. Specifically, the extracted text from images needs to be checked before it get fed into a text classification part of the OCR-based detectors. Although the recognition accuracy of OCRs has been improving over the last years in document classification tasks, these accuracy drops with noisy images. Thus, improving the text recognition accuracy and robustness of OCRs in an adversarial setting is an area that requires more research. Additionally, another area that future research needs to focus on is the word order of the extracted text from an image. One of the drawbacks of OCRs when used for detecting spam images is that the embedded text gets extracted as a list of words, which affects the text classification. Hence, finding a method for re-ordering the extracted words need to be considered.

Chapter 7

Conclusion

This chapter discusses the work accomplished by this thesis, and it highlights the major contributions. Also the revision of the research questions and limitations of our studies are presented. Finally, the future work with recommendations are addressed in this chapter.

7.1 Major Contributions

While ML has been widely adopted to automate the detection of malicious messages (e.g., spam) in Twitter and other OSNs, the research efforts on investigating the vulnerability of this type of detection systems in an adversarial environment are very limited. This is especially true for the detection of spam in Twitter trending hashtags, where it is difficult to use traditional security techniques (e.g., encryption) to limit adversaries' access to the system, since both legitimate users and adversaries have the same access channel. Twitter trending hashtags are preferable places for adversaries due to the large number of participants, which help adversaries to quickly spread their spam messages and reach a large number of victims. This thesis demonstrate the importance of designing adversary-aware ML-based detectors of Twitter spam. The study of the health-related spam campaigns in Twitter Arabic hashtags shows that some spam tweets need to be treated as adversarial examples because they could be used not only to evade detection (*i.e., exploratory attacks*) but also to undermine the deployed ML-based detector (*i.e., causative attacks*). Existing works indicate that any solution designed for an adversarial environment should consider the presence of adversaries and the adaptability to emerging threats. We add that human-interpretablity also needs to be considered so that a designated security analyst could easily monitor and debug the deployed ML-based system. In this thesis, we emphasize the importance of human involvement in ML-based cybersecurity systems and demonstrate how to incorporate human assistance into the ML's final decision. The following paragraphs summarize the major contributions of the thesis.

Identifying Possible Adversarial Attacks. In Chapter 3, we propose the first taxonomy of possible adversarial attacks in Twitter hashtags. Although there exist general taxonomies of possible adversarial attacks against ML, we believe that it is important to identify possible adversarial attacks against Twitter ML-based detectors since attack scenarios are an application-specific issue. To do so, we use ongoing spam campaigns that spread health-related advertisements on Twitter Arabic hashtags as a case study throughout the thesis. We demonstrate how some tweets posted by these

campaigns can be used as adversarial examples to undermine or evade detection. In the proposed taxonomy, we present several possible attack and potential defense strategies. To map the identified attacks and proposed detectors to the proposed taxonomy, the identified attacks (using hijacked accounts, and manipulating images' textual content) fall under *Exploratory Integrity Attack*, in which adversaries' aim is to evade being detected by the deployed classifier without any direct influence on the training data. The defence approach considered for designing the proposed solutions was disinformation, where the aim is to hide some of the important information about the system from an adversary. We use MCS to make it difficult for adversaries to compromise the detectors.

An Adversary-aware ML-based Detector. In Chapter 4, we continue investigating the targeted spam campaigns to find possible adversarial attacks against traditional ML-based detection of Twitter spam. After performing an empirical analysis of the targeted campaigns, we found that they use a unique type of hijacked accounts to fool existing ML-based detectors. Consequently, we develop an adversary-aware ML-based detector, in which we designed a new feature that improves the robustness of the detector to the identified adversarial hijacked accounts. Also, our adversary-aware ML-based detector was designed to be adaptable and interpretable by using multiple classifiers with human-in-the-loop approach. Although we focus on detecting spam posted by health-related ads campaigns, the new designed feature can detect any hijacked account that has characteristics similar to the identified ones. We have found several spam campaigns (see Figure 4.30) using similar hijacked accounts, such as pornography and religious propaganda campaigns.

An Approach for Detecting Adversarial Spam Image. In Chapter 5, we develop an adversary-aware OCR-based detector of Twitter spam images after finding that the targeted spam campaigns used a substantial number of adversarial spam images. The OCR-based detector was designed not only to be robust to adversarial spam images, but also to detect evolving attacks, as we found that some spam images contain manipulated text. Although a few spam image detectors have been proposed before, the robustness to images with embedded manipulated Arabic and English text has not been studied. Also, the adaptability and interpretability of OCR-based detectors have not been considered in related studies. Additionally, the proposed detector can be generalized for different social networking platforms as well as different spam campaigns since it has not only designed for detecting spam images in Twitter. It was designed to extract text from any type of images and classify the extracted text. Moreover, it was designed to detect spam images with embedded English and Arabic text.

An OCR Post-correction Algorithm. In Chapter 6, we extend the adversarial attack carried out by the health-related spam campaigns, in which they purposely manipulate the spam words embedded in images to fool spam image detectors. We develop a black-box method for generating images of manipulated embedded text. Then, we propose an OCR post-correction algorithm for improving the robustness of OCR-based detectors to adversarial text images by denoising and classifying adversarial embedded text in images. Also, we developed an adversary-aware OCR-based detector, in which the proposed algorithm was used, and a text classification model was designed to ensure the adaptability and interpretability of the detector. Similar to the proposed spam image detector in the previous chapter, this detector can be generalized for different social networking platforms as well as different spam campaigns. Although we designed our detector to be robust against a particular type of adversarial attacks, where adversary replaces characters with their similar looking symbols

or numbers, our detector could be modified to detect different adversarial attacks. For example, it could be modified to detect an adversarial text attack, in which adversaries replace some characters in important words with asterisks (e.g., st**id). This has been shown in Section 6.6.

7.2 Review of Research Questions

All research questions of this thesis that were introduced in Section 4.1 have been addressed. The three research questions are as follows:

- **RQ1** *What are the possible adversarial attacks on Twitter trending hashtags?*

  In Chapter 3, after studying the characteristics of the targeted spam campaigns' messages, we identified some spam tweets that can be used as adversarial examples. In Chapter 4, the qualitative analysis of the datasets collected from Twitter Arabic hashtags helps us to identify a new adversarial attack. Also, the experiments in Chapters 5 and 6, show how two new adversarial attacks can be performed in practice.

- **RQ2** *Are existing ML-based detectors of Twitter spam vulnerable to adversarial examples?*

  As shown in the experiments of Chapters 4 , 5 and 6, existing ML-based spam detectors are vulnerable to the identified adversarial examples, as they have not been designed for adversarial settings. Specifically, the evaluation of existing hijacked account detectors in Chapter 4 shows that they cannot detect the identified Adversarial Hijacked Account reliably because of the unique characteristics of these accounts. Additionally, in Chapters 5 and 6, we show how the targeted spam campaigns manipulate the text embedded in images to mislead existing OCR-based detectors.

- **RQ3** *How can we design adversary-aware ML-based detectors of spam in Twitter trending hashtags?*

  Throughout the thesis, we show that designing adversary-aware detectors of Twitter spam can be achieved by considering the three key points: robustness, adaptability and interpretability. The experiments in Chapters 4 , 5 and 6 show that our proposed adversary-aware detectors are robust to the identified adversarial examples, are adaptabile to evolving attacks and interpretable to security analysts. We show that designing a robust feature helps robustifying existing hijacked accounts detectors. Also, adding a OCR-post correction step to OCR-based detectors improves their robustness to adversarial text images. Additionally, we show that using MCS including a FRB with HITL approach makes the proposed adversary-aware detectors adaptable to emerging attacks and human-interpretable for debugging.

7.3 Limitations

We briefly summarise the limitations of each chapter in this thesis and point out some promising research directions that researchers can consider. These limitations are listed bellow:

Chapter 4 introduces *avg_posts*, a new designed feature for detecting the identified Adversarial Hijacked Account. We show that this feature is faster to compute than features used in the literature

and it has the greatest impact on the deployed ML algorithm when making predictions. While the new feature improves the robustness to the adversarial hijacked accounts, it cannot detect other types of hijacked accounts that have different characteristics than the identified ones, with the same level of accuracy. For example, we found that some hijacked accounts have a total number of posts over the account's lifetime that is similar to a legitimate account. However, our experiments show that the designed feature *avg_posts* does not affect the overall accuracy of the deployed ML algorithms as they do not make predictions solely based on the new feature.

Chapter 5 introduces an approach for detecting Adversarial Spam Image posted by the targeted spam campaigns. One of the lessons learnt from this study is that training the text recognition part of the OCR is harder than the text localization part. Building a text recognition model from scratch requires a large number of training samples with high quality. Another lesson is that when training a test recognition model using different languages, the model fails to recognize some litters that have similar shapes. For instance, the letter (و) in Arabic is recognized as the letter (Q). However, we build separate models for English and Arabic languages to overcome this issue.

Chapter 6 presents an adversary-aware OCR-based detector of Twitter spam images. This detector has been designed to be robust against Adversarial Text Image that could be posted by the targeted spam campaigns. In this research, an OCR post-correction algorithm was proposed to improve the robustness. In order for our algorithm to be effective, the OCR system needs to accurately recognize the embedded text in images. The recognition accuracy of OCRs gets affected by noise and adversarial perturbations. This is especially the case for images uploaded into OSNs, which are images with text overlaid on top of them. Extracting text from such images with high accuracy is an area that requires more research. Also, the robustness of the OCR-based systems to adversarial image attacks (e.g., FGSM or Adversarial Watermarks) needs to be considered. Additionally, one of the drawbacks of OCR systems when used for detecting spam images is that the embedded text gets extracted as a list of words, which affects the text classification. In this study, we used an MCS with FRB to tackle this issue. However, more research to find a method for re-ordering the extracted words needs to be considered.

Through out the thesis, we use MCS to analyse different features of Twitter's messages and ensure the adaptability and interpertibility of our proposed detectors. Although it has been proven in the literature that MCS improves the accuracy and security, there are some limitations of this approach. For example, the execution time, computational complexity, and memory storage. Each algorithm used in a MCS has different execution time, which could increase the computational complexity and memory storage. Generally, in classification tasks, speed and accuracy are important criterion, which are competing against security. In this thesis, we focus on the security of ML-based models by using MCS, and investigating the limitations of MCS, will be left for future work.

Experiments preformed in this thesis consist of potential threats to validity. Regarding the internal validity, a potential threat involves the choice of datasets. To mitigate this threat, the developed datasets were labeled by two other annotators, and the IAA test was used. Also, other benchmark datasets that are widely used in the literature were used in the experiments. However the size and the types of adversarial examples in datasets may influence the results. Another potential threat is weather the improvement in the detection accuracy of the investigated models under the identified adversarial attacks has affected by factors other than the newly designed feature,

adversarial training and proposed algorithm. To mitigate this threat, we have not evaluated the importance of the new designed feature in detecting the identified hijacked accounts by using a specific ML model or a particulate test. Instead, different traditional ML, DL models and three feature selection methods were used to evaluate the importance. Also, the proposed OCR post-correction algorithm was evaluated by using three SOTA OCRs. Additionally, the robustness of the proposed algorithm was evaluated against four text manipulation attacks and the results were compared with five spell checking tools.

Regarding the external validity, a possible threat is related to the generalization of the results. Although most of the datasets used for experiments were collected from Twitter Arabic hashtags (i.e., health-related campaigns), datasets collected from different platforms (e.g., SMS and Wikipedia) were also used. The experiments' results in chapter 4, show that when using datasets that have different underlying distributions, the overall accuracy were affected, but the recalls (i.e., the ability to detect malicious samples) have improved. This is understandable as the the new feature were designed to detect the identified hijacked accounts. Thus, results might be different on datasets that have different underlying distributions. However, our findings have shown that some other campaigns were found to be using hijacked accounts that have similar characteristics to the identified ones, so the designed feature can be generalised for other campaigns. Additionally, the characteristics of the identified hijacked accounts can be used to craft adversarial examples for different OSNs. Moreover, the developed adversary-aware spam image detectors can be used for classifying any types of images since they were designed to extract text from any types of image and were evaluated and compered to different types of OCRs (Facebook, and Google Vision).

Regarding the construct validity, we follow evaluation methods used in related studies. We did not preformed our experiments on the collected datasets exclusively, instead, we used datasets that are widely used in related studies. The adopted models were trained on clean training datasets and evaluated by using testing datasets that contains adversarial examples. These validation techniques were applied to evaluate the robustness of the adopted models, and the detection accuracy, recall and precision were used as evaluation metrics. Another potential threat is whether the results of the evaluated models would be affected when increasing the sample size of the datasets. Although the size of the collected dataset is not large, experiments show that a small size of adversarial examples affect the performance of the adopted models. Thus, increasing the size of the datasets will not affect our results.

7.4 Lessons Learned

Considering the three key points: robustness, adaptability, and interpretability when designing an adversary-aware ML-based model for cybersecurity systems is important. Cybersecurity domains are adversarial environments, so any designed solution needs to take into account the presence of an active adversary and needs to evolve over time in the face of new adversarial activities. Throughout this thesis, different lessons are learned while designing and evaluating the proposed adversary-aware detectors.

Robustifying ML-based Models. There are two methods that commonly used for robustifying ML-based models: detection/rejection or robust optimization (a.k.a. adversarial training). We

have investigated the use of both methods for improving the robustness of Twitter spam detectors in trending hashtags. The learning process of ML-based detectors involve three main steps: data collection, feature extraction/ engineering, and training and testing an ML algorithm. The accuracy of detection systems depends heavily on the engineered features. An ML-based detector designed with simple or weak features in an adversarial environment, where an active adversary continuously changes their behaviours over time to avoid detection, can become obsolete quickly. An important step that is often used to mitigate spammers' evasion tactics (i.e., adversarial attacks) is designing new and more robust features to detect/ reject these attacks. A robust feature is a type of features that is either difficult or expensive for the adversaries to evade [278]. Thus, one of the lessons learned while evaluating and improving Twitter spam detectors' robustness to adversarial examples is building robust features. This step is often overlooked in the literature as most studies focus only on the detection accuracy. Additionally, our work of using adversarial training to improve the robustness of spam image detectors supports existing studies' results that this defensive method can improve the robustness but requires a dataset with sufficient adversarial examples. Also, this defence method is highly effective against known attacks, but it is expensive and time consuming. Consequently, we proposed an extra step to OCR-based detectors to improve the robustness through detection/rejection type of defence.

Adaptability and Interpretability of ML-based Models The arm race between adversaries and defenders is a man to man game, so designing a fully automated system as a defensive method is not a reliable approach to solve cybersecurity issues. Thus incorporating human (e.g., a security analyst or designer) not only for the data collection and annotation tasks, but to aid the final decision of the ML-based detection system is important. Human intervention to ML-based detectors is inevitable, so the question is when and how to effectively incorporate human supervision into the detection process. An important lesson that we learned when designing our adversary-aware detectors is that the they should not only be robust against the identified adversarial attacks, but also they need to adaptable to handle new emerging attacks (e.g., adversarial drift), and interpretable to a security analyst. The incorporation of human to aid the final decision of ML-based detectors plays an important role to ensure that the model can evolve over time.

Current studies in detecting Twitter spam use an MCS detection systems and a softmax layer or ML algorithm for aggregating the output of the classifiers. Such aggregating methods are perceived as black-box, so it is hard for a security analyst to interact with them for debugging. However, we use MCS with a FRB, which is more interpertable than the black-box models used by related studies, for aggregating the results of the multiple classifiers and to enable the security analyst (HITL approach) to interact and debug the model. Since detecting spam in Twitter requires analysing different types of massages' features, we focus on the interpretability of ML-based models (i.e.,*global interpretability*) instead of the interpretability of the model's components (i.e., *local interpretability*). We believe that at the design stage, it is important to consider the global interpretability of the ML-based model and the local interpretability at the debugging stage. To ensure the adaptability of ML-based models, we need to be able to detect attacks and know which part of the model is affected. Using the FRB that include few rules can help the security analysis with these tasks. Once an attack is detected, the internal interpretability of the model can help with the debugging task.

7.5 Future Work

The aim of this thesis was to design adversary-aware ML-based detectors of Twitter spam that are robust, adaptable and interpretable. The analysis of the targeted spam campaigns in Twitter Arabic hashtags revels a number of possible adversarial attacks that can be carried out by these campaigns. This theses investigates some of these adversarial attacks; in particulate evasion type of attack that can be launched at testing stage. Moving forward, we hope to deepen this investigation, extending our works to a causative type of adversarial attacks, where the adversaries can influence the deployed spam detector at the training stage. Initially, we will focus on the following three research directions.

Poisoning Attacks. The analysis of the targeted spam campaigns on Twitter trending hashtags shows that adversaries flood hashtags with spam images to undermined the deployed spam detectors and launching a more complex attack. Defending against such attacks in Twitter hashtags is challenging due to the fast flow of messages. These types of attacks affect the availability of ML-based detectors and users' experience. Hence, studying poisoning attacks against ML-based detectors of spam in Twitter hashtags is a potential research direction.

Detecting Spam Videos. Another potential research direction is analysing videos attached to Twitter's messages. After analysing the targeted spam campaigns on Twitter trending hashtags, we found that they use some spam messages that include videos. The detection of spam videos on Twitter has not been well studied.

Causative Types of Attacks. Concerns about the vulnerability of ML-based detectors to adversarial attack at training stage have been raised recently. Thus, investigating such type of attack on Twitter Trending Hashtags is needed. The training of ML algorithms when designing Twitter spam detectors is often done by using publicly available datasets. These publicly available datasets can be contaminated by adversaries.

The integration of spam and malware detection. Since spam massages in OSNs are widely used to distribute malware, integrating spam and malware detection is an interesting direction for future work. Spammers can craft spam tweets that include malicious URL links or images to lure users into clicking/opening them. The study of the targeted spam campaigns in this thesis, show that they use a large number of spam images. These images could be used by these campaigns to disseminate malware. Hence, adapting the proposed adversary-aware detectors to detect malware through analysing tweets' content (e.g., URLs and images) is a crucial area to explore.

Appendix A

User Study Survey

Do you agree to participate in this study?

- Agree
- Disagree

What is your gender?

- Male
- Female
- Other
- Prefer not to answer

What is your age?

- Under 18
- 18 24
- 25 34
- 35 44
- 45 54
- 55 64
- 65 74
- 75 84
- 85 or older

What is your level of education?

- Less than high school

- High school graduate
- Some college
- Bachelor degree
- Master degree
- PhD

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- holland
- holiday
- hollywood
- hold

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- chot
- chat
- ch 1
- cheat

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- massage
- message
- me and age
- messenger

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- 1 test
- latest
- a test
- test

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- Buy
- Busy
- Bussy
- Buss

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- Coller
- Caller
- Celler
- Call

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- ringtone
- right one
- ringtail
- ringing

Can you understand text on this image?

- Yes

- No

Please choose the text that best matches the text in the image?

- calm
- call me
- claim
- ciara

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- we ordered
- ordered
- awarded
- award

Can you understand text on this image?

- Yes
- No

Please choose the text that best matches the text in the image?

- 1 and line
- landline

- I am in
- land in

Have you ever seen similar images?

- Yes
- No

In your opinion, where this kind of images could be found?

- Email
- Social Media
- SMS
- Newspaper
- All of the above

Do you use scrambled text (i.e words consist of a combination of letters and symbols) when you write a text?

- Yes
- No

Appendix B

Twitter Spam Detectors

Table 2.1 and 2.2 present comparison of features and methodologies used for detecting hijacked accounts in related Work. Also, Table 2.3 shows an outline of recent spam detectors proposed in the literature.

Table 2.1: Comparison of Features and Methodologies Used in Related Work

#	Models	Features	Methodology	Acc	Remarks
1	Detecting Hacked Twitter Accounts based on Behavioural Change	Frequency, Time, source, Language, Retweet, URL	They build behaviour profiles (e.g language, sources) for each user and compare the value of new tweet features to distinguish compromised and normal tweets. - Accounts must have at least 10 posts to be able to build a behaviour profile. - Real-world dataset of hijacked accounts.	Accuracy 99.35	Computationally expensive to build profile for every users' account. Some of the hijacked accounts on our dataset contain fewer than 10 posts. Some hijacked accounts only contain spam posts.
2	Towards detecting anomalous user behavior in online social networks	Temporal Features, Spatial Features	They use PCA to encode user-level features and decode them using reconstruction error to detect compromised accounts	Accuracy 66	It requires a large sample size to build users' behaviour profile. Computationally expensive to build profile for every users' account
3	COMPA	Topic, URL repetition, Source, Time, Language, Proximity, user interaction	They build behaviour profiles (e.g.,posting, retweeting) for accounts and then detects anomalies. (user-level detection). Accounts must have at least 10 posts to be able to build behaviour profile. Real-world dataset of hijacked accounts.	Analysis only	- Computationally expensive to build a profile for every users' account. Not applicable for early detection. Can't detect accounts with few posts or that only have spam posts.
4	Evaluating Algorithms for detection of compromised social media user accounts	Time, source, hashtags, language, links, mentions	An improved version of COMPA. Dataset with synthetic compromised accounts. They build behaviour profiles for a set of accounts, and evaluate each new post. They use dataset with synthetic compromised accounts.	Analysis only	- Computationally expensive to build a profile for every users' account. Not applicable for early detection. Can't detect accounts with few posts or that only have spam posts.
5	Understanding Compromised Accounts on Twitter	Is Retweet, Sentiment, Source, Text, URL	They analyze 150 tweets of each account by extracting tweet-level and user-level features. Real-world dataset of hijacked accounts.	Analysis only	- It requires a large sample size (150 tweets) to build users' behaviour profile. Computationally expensive to build profile for every users' account.
6	End-to-End Compromised Account Detection	Text, URL, Hashtags, Mentions, Language, Time of Day, Source, Location, Is Retweet, Is sensitive, Media, Coordinates, Norm. Neighbors, LCC	They use all users' posts to learn a user representation. The framework captures tweet-level, user-level, and network-level features. The best results achieved when using user-level and network-level features. Real-world dataset of hijacked accounts.	Recall 0.6	The performance of the framework is highly dependent on the user-level features. It uses all users' posts to determine whether the account is compromised or not. It requires a large sample size to build users' representation.
7	CADET: A Multi-View Learning Framework for Compromised Account Detection on Twitter	Term, Source, Time, Location	It uses a combination of textual and metadata features. They rely on sources commonly used by compromised accounts, time of the day compromised accounts posts at, and compromised accounts' frequent locations. Real-world dataset of hijacked accounts.	Precision 0.58	- It requires a large sample size (150 tweets) to build users' behaviour profile. Some of the hijacked accounts that we found have very few total number of posts (e.g., 2 or 3). Some hijacked accounts only contain spam tweets (i.e., do not contain the original account's tweets), so textual-based classifiers can't detect them.
8	You have been CAUTE! Early Detection of Compromised Accounts on Social Media	Hashtags, Mentions, URLs, Text, Time of Day, Language, Source, location	It considers a tweet lexical and meta features. They use two encoders: one transforms tweet features to user features and the other predicts the content of a tweet from the user's features and meta-features. It uses a residual2class classifier for the final decision. They extract features from the first k% (e.g., 10) tweets of an account. They use dataset with synthetic compromised accounts.	AUC 0.70	For this framework to be effective, at least 40% of users' tweets need to be extracted to build users' tweets representation. However, some of the hijacked accounts that we found have very few total number of posts (e.g., 2 or 3). Also, some hijacked accounts only contain spam tweets (i.e., do not contain the original account's tweets), so textual-based classifiers can't detect them.

Table 2.2: Comparison of Features and Methodologies Used in Related Work

#	Models	Features	Methodology	Acc	Remarks
9	Identifying compromised accounts on social media using statistical text analysis	Text	They group users' messages into users' messages and spam' messages and measure the similarities between the two groups using KL-divergence. (user-level detection). They use dataset with synthetic compromised accounts.	Recall 0.61	It requires a large sample size (150 tweets) to build users' behaviour profile. Some of the hijacked accounts that we found have very few total number of posts (e.g., 2 or 3) and others only contain spam tweets (i.e., do not contain the original account's tweets)
10	Semantic Text Analysis for Detection of Compromised Accounts on Social Networks	Text	They divide accounts' tweets into regular and compromised and measure the difference between words. They use KL-divergence to measure the distance between words. They use dataset with synthetic compromised accounts	Recall 0.68	It requires a large sample size (150 tweets) to build users' behaviour profile. Some of the hijacked accounts that we found have very few total number of posts (e.g., 2 or 3) and others only contain spam tweets (i.e., do not contain the original account's tweets)
11	UbCadet: detection of compromised accounts in twitter based on user behavioural profiling	Tweet, hashtag, time, language, country code, source, URL, retweet	They use these features to build a user behavioural profile for each user. Real-world dataset of hijacked accounts.	Accuracy 22.0	Computationally expensive to build a profile for every users' account. Not applicable for early detection. Can't detect accounts with few posts or that only have spam posts.
12	Our Model	Text, Hashtags, Followers, Friends, Lists, Source, Reputation, words, characters, URLs, Status, Account age, avg. posts, Description, Emoji	It considers tweet content and meta-feature to capture spam. It uses a traditional ML-based, neural network-based and a rule-based for the final decision. It detects tweets posted by hijacked accounts at tweet-level.	Recall 0.73	- Unlike the above models, It is applicable for early detection since it doesn't require analysing users' posts history. The new feature can be used to reduce the computational time of the above models. For example, instead of checking every user's accounts, they can check accounts that have a lower avg_posts. None of the above consider the usage of hijacked accounts as an adversarial attack. None of the above consider detecting hijacked accounts in streaming environments. Most of the previous studies use datasets of synthetic compromised accounts that have at least 100 posts.

Table 2.3: Outline of some recent techniques used for detecting spam in Twitter: some of these works are discussed in this Section 2.1

Title	Methodology	Type of Spam	Type of Detector	Learning Approach	Results/Accuracy
6 Million Spam Tweets - A Large Ground Truth for Timely Twitter Spam Detection [62]	Different ML algorithms were used; balanced and imbalanced datasets were tested.	Spam tweet	Feature-based	Supervised	RF outperforms other algorithms.
A Hybrid Approach for Spam Detection for Twitter [190]	J48, Decorate and Naive-Bayes (NB).	Spam tweet	Feature-based,user-based,and graph-based	Supervised	J48 outperforms other algorithms.
Leveraging Time for Spammers Detection on Twitter	Time-based features were used, and different ML algorithms were tested.	Spam tweet	Feature-based	Supervised	RF outperforms other algorithms.
Leveraging Time for Spammers Detection on Twitter [274]	Time-based features were used, and different ML algorithms were tested.	Spam tweet	Feature-based	Supervised	RF outperforms other algorithms.
Twitter spam detection based on Deep Learning [275]	Different ML algorithms with Word2Vector technique were used.	Spam tweet	Syntax-based	Supervised	RF with Worrd2Vec outperforms other algorithms.
Semi-supervised spam detection (S3D) [222]	Utilizes four lightweight detectors (supervised and unsupervised) to detect spam tweets and updates the models periodically in batch mode.	Spam tweet	Feature-based and Blacklist	Semi-supervised	Confidential labeling process, which uses blacklisted, near-duplicated, and reliable non-spam tweets, makes the deployed classifier more efficient when detecting new spam tweets.
CrowdTarget: Target-based Detection of Crowdturfing in Online Social Networks [242]	Detects spam tweets that received retweets from malicious crowdsourcing customers. It uses four new retweet-based features and KNN as a classifier.	Spam tweet	Feature-based	Supervised	CrowdTarget detects malicious retweets created by crowdturfing users with a True Positive Rate of 0.98 and False Positive Rate of 0.01.
Beating the Artificial Chaos - Fighting OSN Spam using Its Own Templates [293]	Detects template-based spam, paraphrase spam, and URL-based spam.	Spam campaign	Syntax-based	Supervised	Template-detection outperforms URL blacklist-detection.
POISED - Spotting Twitter Spam Off the Beaten Paths [206]	Detects spam campaign based on community and topic of interest.	Spam campaign	Syntax-based	Supervised and unsupervised	NB, SVM, and RF all achieve about 90% detection accuracy.
Collective Classification of Spam Campaigners on Twitter: A Hierarchical Meta-Path Based Approach (HMPS) [122]	Builds heterogeneous networks and detects nodes connected by the same phone number or URL.	Spam campaign	Social graph-based	Supervised	HMPS outperforms feature- and content-based approaches. Prediction accuracy improves when using HMPS with feature- and user-based approaches.
Social Fingerprinting - Detection of Spambot Groups Through DNA-Inspired Behavioral Modeling [72]	Models users' behaviour using DNA fingerprinting technique to detect spambots.	Spam campaign	Social graph-based	Supervised and unsupervised	The results show that the proposed approach achieves better detection accuracy when using a supervised learning approach.

Appendix C

Adversarial Machine Learning

Tables 3.1 and 3.2 outline recent works in adversarial machine learning.

Table 3.1: Outline of different adversarial attacks

Type of Influence	Title	Name of Attack	Attack Target	Attack Method
Causative	Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning [145]	Poisoning	Regression Learning	Optimization-based poisoning attack, in which different optimization approaches were used. Statistical-based poisoning attack (StatP) that queries a deployed model to find an estimate of the mean and covariance of training data.
	Support vector machines under adversarial label noise [34]	Label Flipping	SVM	Two different label flipping attacks were used: random and adversarial label flips.
	Curie - A method for protecting SVM Classifier from Poisoning Attack [163]	Label Flipping	SVM	Two label flipping attack were used. In the first, the loss maximization framework was used to select points that needed their label to be flipped. In second attack, the selected data points are moved to other points in the feature space.
	Adversarial Machine Learning [129]	Dictionary	Spam filter	An adversary builds a dictionary of tokens learned from the targeted model, and then sends attack messages to cause misclassification.
	Thwarting Signature Learning by Training Maliciously [205]	Red Herring	Polymorphic worm signature generation algorithms	An adversary sends messages with fake features to trick the deployed model.
	Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers [269]	Poisoning	NB, BN, SVM, J48, RF	Two types of poisoning attacks were performed: Injecting misleading samples and altering training data.
Exploratory	Data Driven Exploratory Attacks on Black Box Classifiers in Adversarial Domains [225]	Anchor Points (AP) and Reverse Engineering attacks (RE)	SVM, KNN, DT, RF	AP attack is not affected by the chosen model (linear or non-linear), unlike RE, which is affected when a defender uses DT or RF.
	Evasion Attacks against Machine Learning at Test Time [36]	Evasion	SVM, Neural Network	A gradient-descent evasion attack was proposed.
	Good Word Attacks on Statistical Spam Filters [184]	Good Word	NB, Maximum entropy filter	Active and passive good word attacks against email spam filters were evaluated.
	Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering [6]	Reverse Engineering	SVM	Three different query selection methods, which help learn the decision boundary of deployed classifier, were used. Random, selective, and uncertainty sampling.
	Man vs. Machine: Practical Adversarial Detection of Malicious Crowdsourcing Workers [269]	Evasion	NB, BN, SVM, J48, RF	Two evasion attack were launched: Basic evasion attack and Optimal evasion attack, where an adversary knows features Needs to be altered.

Table 3.2: Outline of techniques used for mitigating adversarial attack

Type of Influence	Title	Name of Attack	Type of Classifier	Defense Category	Defense Method
Causative	Mitigating Poisoning Attacks on Machine Learning Models: A Data Provenance Based Approach [19]	Poisoning	SVM	Data Sanitization	Filtering out poisoned data from the training dataset using a provenance framework that records the lineage of data points.
	Curie- A method for protecting SVM Classifier from Poisoning Attack [163]	Poisoning	SVM	Data Sanitization	The data are clustered in the feature space, and the average distance of each point from the other points in the same cluster is calculated, with the class label considered as a feature with proper weight. The data points with less than 95% confidence are removed from the training data.
	Bagging Classifiers for Fighting Poisoning Attacks in Adversarial Classification Tasks [35]	Poisoning	Bagging and weighted bagging ensembles	Data Sanitization	Using an ensemble construction method (bagging) to remove outliers (adversarial samples) from training dataset.
	Data sanitization against adversarial label contamination based on data complexity [57]	Label Flipping	SVM	Data Sanitization	Data complexity, which measures the level of difficulty of classification problems, was used to distinguish adversarial samples in the training data.
	Support vector machines under adversarial label noise [44]	Label Flipping	SVM	Robust learning	Adjusting the kernel matrix of SVM depending on noise (adversarial) samples' parameters increases the robustness of the classifier.
	Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning [145]	Poisoning	Regression Learning	Robust learning	The TRIM algorithm, which regularized linear regression by applying trimmed optimization techniques, was proposed
Exploratory	Robust support vector machines against evasion attacks by random generated malicious samples	Evasion	SVM	Robust learning	Trains the SVM classifier with random malicious samples to enclose the decision function.
	Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering [6]	Reverse Engineering	SVM	Randomization	Learning a distribution of classifiers and picking a decision boundary randomly makes reverse engineering attacks harder to launch.
	Handling adversarial concept drift in streaming data [227]	Evasion	SVM	Disinformation	Hiding the importance of features and using an ensemble of classifiers.
	Adversarial Pattern Classification using Multiple Classifiers and Randomization [38]	Evasion	Spam Filter, SVM, NB	Multiple Classifiers and Randomization	Multiple Classifiers Strategy MCS, where different classifiers are trained by different features to randomize a model's decision boundary.

Glossary

adversarial attack a type of attacks, where adversaries crafted adversarial examples either to evade detection or to undermine the performance of the detector model. 20
20

Adversarial Spam Image A spam image with embedded manipulated text. 5
5, 7
7, 95
95, 132
132

Adversarial Hijacked Account An inactive hijacked account used to fool Ml-based spam detectors. 5
5, 6
6, 131
131

Adversarial Machine Learning a research field that investigates the vulnerability of ML to adversarial examples. 2
2, 20
20

Adversarial Text Image An image with embedded manipulated text. ii, 5
5, 7
7, 132
132

adversary-aware a model which is designed by taking into account the presence of an adversary. ii

twitter hashtag A specific word or group of words, which preceded with the"#" symbol, allow communities and discussions to grow around a topic. 1
1, 21
21, 24
24, 47
47

Acronyms

BERT Bidirectioal Encoder Representations from Transformers. 110
110

CNN Convolutional Neural Network. 94
94

CRNN Convolutional Recurrent Neural Network. 98
98

CTC Connectionist Temporal Classification. 94
94

DL Deep Learning. ii, 2
2, 23
23

DNN Deep Neural Network. 94
94

FRB Fuzzy Rule Based. 54
54, 63
63, 65
65, 83
83, 91
91, 122
122, 131
131, 134
134

HITL human-in-the-loop. ii, 7
7, 65
65, 86
86, 131
131, 134
134

RF Random Forests. 2
2, 23
23, 71
71

SOTA State-Of-The-Art. ii, 6
6, 7
7, 8
8, 17
17, 66
66, 74
74, 78
78, 96
96, 101
101, 102
102, 108
108, 118
118, 120
120, 124
124, 127
127, 133
133

SVM Support Vector Machine. 2
2

URL Uniform Resource Locator. 1
1, 2
2

Bibliography

[1] Lida Abdi and Sattar Hashemi. To combat multi-class imbalanced problems by means of over-sampling and boosting techniques. *Soft Computing*, 19(12):3369–3385, 2015.

[2] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, pages 559–560, 2018.

[3] Saad Bin Ahmed, Saeeda Naz, Muhammad Imran Razzak, and Rubiyah Yousaf. Deep learning based isolated arabic scene character recognition. In *2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, pages 46–51. IEEE, 2017.

[4] Nora Al Twairesh, Mawaheb Al Tuwaijri, Afnan Al Moammar, and Sarah Al Humoud. Arabic spam detection in twitter. In *The 2nd Workshop on Arabic Corpora and Processing Tools 2016 Theme: Social Media*, page 38, 2016.

[5] Ala' M Al-Zoubi, Ja'far Alqatawna, and Hossam Faris. Spam profile detection in social networks based on public features, 2017.

[6] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 231–240, 2014.

[7] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 231–240, 2014.

[8] Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Adding Robustness to Support Vector Machines Against Adversarial Reverse Engineering. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 231–240, 2014.

[9] Al-Zoubi Ala'M, Ja'far Alqatawna, and Hossam Faris. Spam profile detection in social networks based on public features. In *2017 8th International Conference on information and Communication Systems (ICICS)*, pages 130–135. IEEE, 2017.

[10] Nuha Albadi, Maram Kurdi, and Shivakant Mishra. Hateful people or hateful bots? detection and characterization of bots spreading religious hatred in arabic social media. August 2019.

[11] Tiago Almeida, José María Gómez Hidalgo, and Tiago Pasqualini Silva. Towards sms spam filtering: Results under a new dataset. *International Journal of Information Security Science*, 2(1):1–18, 2013.

[12] Tiago A Almeida, José María Gómez, and Akebo Yamakami. Contributions to the study of SMS spam filtering: New collection and results.

[13] Basemah Alshemali and Jugal Kalita. Toward mitigating adversarial texts. *International Journal of Computer Applications*, 178(50):1–7, 2019.

[14] Plamen P Angelov and Xiaowei Gu. Deep rule-based classifier with human-level performance and characteristics. *Information Sciences*, 463:196–213, 2018.

[15] Annapurna Annadatha and Mark Stamp. Image spam analysis and detection. *Journal of Computer Virology and Hacking Techniques*, 14(1):39–52, February 2018.

[16] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[17] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4715–4723, 2019.

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[19] Nathalie Baracaldo, Luis Angel D Bathen, Roqeeb O Ozugha, Robert Engel, Samir Tata, and Heiko Ludwig. Securing data provenance in internet of things (IoT) systems. In *Service-Oriented Computing – ICSOC 2016 Workshops*, pages 92–98. Springer International Publishing, 2017.

[20] Sasan Barak, Azadeh Arjmand, and Sergio Ortobelli. Fusion of multiple diverse predictors in stock market. *Information Fusion*, 36:90–102, 2017.

[21] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J D Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, 2010.

[22] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J D Tygar. The security of machine learning. *Mach. Learn.*, 81(2):121–148, 2010.

[23] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J D Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 16–25, New York, NY, USA, 2006. ACM.

[24] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25, 2006.

[25] Sukarna Barua, Md Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on knowledge and data engineering*, 26(2):405–425, 2012.

[26] Aliaksandr Barushka and Petr Hajek. Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. *Applied Intelligence*, 48(10):3538–3556, October 2018.

[27] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*, 2017.

[28] M Ben Halima, H Karray, and A M Alimi. Arabic text recognition in video sequences. August 2013.

[29] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on twitter. In *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS)*, volume 6, page 12, 2010.

[30] David M Beskow and Kathleen M Carley. Its all in a name: Detecting and labeling bots by their name. December 2018.

[31] Alessandro Bessi and Emilio Ferrara. Social bots distort the 2016 us presidential election online discussion. *First Monday*, 21(11-7), 2016.

[32] B Biggio and F Roli. Wild patterns: Ten years after the rise of adversarial machine learning half-day tutorial. In *25th ACM Conference on Computer and Communications Security, CCS 2018*, pages 2154–2156. Association for Computing Machinery, 2018.

[33] Battista Biggio. Adversarial pattern classification. *Sardinia: University of Cagliari*, 2010.

[34] Battista Biggio. Support vector machines under adversarial label noise. *JMLR Workshop Conf. Proc.*, 20:97–112, 2011.

[35] Battista Biggio, Igino Corona, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *Lecture Notes in Computer Science*, pages 350–359. 2011.

[36] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[37] Battista Biggio, Giorgio Fumera, Ignazio Pillai, and Fabio Roli. A survey and experimental evaluation of image spam filtering techniques. *Pattern Recognit. Lett.*, 32(10):1436–1446, July 2011.

[38] Battista Biggio, Giorgio Fumera, and Fabio Roli. Adversarial pattern classification using multiple classifiers and randomisation. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 500–509. Springer, 2008.

[39] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2013.

[40] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security Evaluation of PatternClassifiers under Attack. *Knowledge and Data Engineering,* , 26(4):984–996, 2014.

[41] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security Evaluation of PatternClassifiers under Attack. *Knowledge and Data Engineering,* , 26(4):984–996, 2014.

[42] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security Evaluation of PatternClassifiers under Attack. *Knowledge and Data Engineering,* , 26(4):984–996, 2014.

[43] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security Evaluation of PatternClassifiers under Attack. *Knowledge and Data Engineering,* , 26(4):984–996, 2014.

[44] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. In *Asian conference on machine learning*, pages 97–112, 2011.

[45] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. December 2017.

[46] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[47] Bayan Boreggah, Arwa Alrazooq, Muna Al-Razgan, and Hana AlShabib. Analysis of arabic bot behaviors, 2018.

[48] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79, 2018.

[49] Fedor Borisyuk, Albert Gordo, and Viswanath Sivakumar. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 71–79. ACM, July 2018.

[50] Federico Boschetti, Matteo Romanello, Alison Babeu, David Bamman, and Gregory Crane. Improving ocr accuracy for classical critical editions. In *International Conference on Theory and Practice of Digital Libraries*, pages 156–167. Springer, 2009.

[51] Michael Bruckner. Static prediction games for adversarial learning problems. *J. Mach. Learn. Res.*, 13:2617–2654, 2012.

[52] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018.

[53] Michal Bušta, Yash Patel, and Jiri Matas. E2e-mlt-an unconstrained end-to-end method for multi-language scene text. In *Asian Conference on Computer Vision*, pages 127–143. Springer, 2018.

[54] Z Cai, Z He, X Guan, and Y Li. Collective Data-Sanitization for preventing sensitive information inference attacks in social networks. *IEEE Trans. Dependable Secure Comput.*, 15(4):577–590, July 2018.

[55] Giuseppe Casalicchio, Christoph Molnar, and Bernd Bischl. Visualizing the feature importance for black box models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 655–670. Springer, 2018.

[56] Edgard Chammas, Chafic Mokbel, and Laurence Likforman-Sulem. Arabic handwritten document preprocessing and recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 451–455. IEEE, 2015.

[57] Patrick PK Chan, Zhi-Min He, Hongjiang Li, and Chien-Chang Hsu. Data sanitization against adversarial label contamination based on data complexity. *International Journal of Machine Learning and Cybernetics*, 9(6):1039–1052, 2018.

[58] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. Debot: Twitter bot detection via warped correlation. In *ICDM*, pages 817–822, 2016.

[59] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[60] C Chen, J Zhang, Y Xie, Y Xiang, W Zhou, M M Hassan, A AlElaiwi, and M Alrubaian. A performance evaluation of machine Learning-Based streaming spam tweets detection. *IEEE Transactions on Computational Social Systems*, 2(3):65–76, 2015.

[61] Chao Chen, Yu Wang, Jun Zhang, Yang Xiang, Wanlei Zhou, and Geyong Min. Statistical features-based real-time detection of drifted twitter spam. *IEEE Transactions on Information Forensics and Security*, 12(4):914–925, 2016.

[62] Chao Chen, Jun Zhang, Xiao Chen, Yang Xiang, and Wanlei Zhou. 6 million spam tweets: A large ground truth for timely twitter spam detection. In *2015 IEEE international conference on communications (ICC)*, pages 7065–7070. IEEE, 2015.

[63] Chao Chen, Jun Zhang, Yang Xiang, and Wanlei Zhou. Asymmetric self-learning for tackling twitter spam drift. In *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 208–213. IEEE, 2015.

[64] L Chen, Y Ye, and T Bourlai. Adversarial machine learning in malware detection: Arms race between evasion attack and defense. In *2017 European Intelligence and Security Informatics Conference (EISIC)*, pages 99–106, 2017.

[65] Sheng Chen, Haibo He, and Edwardo A Garcia. Ramoboost: Ranked minority oversampling in boosting. *IEEE Transactions on Neural Networks*, 21(10):1624–1642, 2010.

[66] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. In *Proceedings of the IEEE international conference on computer vision*, pages 5076–5084, 2017.

[67] Christy MK Cheung and Matthew KO Lee. A theoretical model of intentional social action in online social networks. *Decision support systems*, 49(1):24–30, 2010.

[68] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Who is tweeting on twitter: Human, bot, or cyborg? *Dec.*, 6:10, 2010.

[69] Zi Chu, Indra Widjaja, and Haining Wang. Detecting social spam campaigns on twitter. In *International Conference on Applied Cryptography and Network Security*, pages 455–472. Springer, 2012.

[70] Eric M Clark, Jake Ryland Williams, Chris A Jones, Richard A Galbraith, Christopher M Danforth, and Peter Sheridan Dodds. Sifting robotic from organic text: a natural language approach for detecting automation on twitter. *Journal of computational science*, 16:1–7, 2016.

[71] Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2(1):1–24, 2015.

[72] S Cresci, R D Pietro, M Petrocchi, A Spognardi, and M Tesconi. Social fingerprinting: Detection of spambot groups through DNA-Inspired behavioral modeling. *IEEE Trans. Dependable Secure Comput.*, 15(4):561–576, July 2018.

[73] Stefano Cresci. Detecting malicious social bots: story of a never-ending clash. In *Multidisciplinary International Symposium on Disinformation in Open Online Media*, pages 77–88. Springer, 2019.

[74] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The Paradigm-Shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 963–972. International World Wide Web Conferences Steering Committee, April 2017.

[75] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th international conference on world wide web companion*, pages 963–972, 2017.

[76] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. Social fingerprinting: detection of spambot groups through dna-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing*, 15(4):561–576, 2017.

[77] Stefano Cresci, Fabrizio Lillo, Daniele Regoli, Serena Tardelli, and Maurizio Tesconi. $ fake: Evidence of spam and bot activity in stock microblogs on twitter. In *Twelfth international AAAI conference on web and social media*, 2018.

[78] Stefano Cresci, Fabrizio Lillo, Daniele Regoli, Serena Tardelli, and Maurizio Tesconi. Cashtag piggybacking: uncovering spam and bot activity in stock microblogs on twitter. April 2018.

[79] Stefano Cresci, Fabrizio Lillo, Daniele Regoli, Serena Tardelli, and Maurizio Tesconi. Cashtag piggybacking: uncovering spam and bot activity in stock microblogs on twitter. *ACM Transactions on the Web (TWEB)*, 13(2):1–27, 2019.

[80] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. From reaction to proaction: Unexplored ways to the detection of evolving spambots. In *Companion Proceedings of the The Web Conference 2018*, pages 1469–1470, 2018.

[81] Stefano Cresci, Marinella Petrocchi, Angelo Spognardi, and Stefano Tognazzi. Better safe than sorry: an adversarial approach to improve social bot detection. In *Proceedings of the 10th ACM Conference on Web Science*, pages 47–56, 2019.

[82] Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 99–108, New York, NY, USA, 2004. ACM.

[83] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

[84] Clayton A Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. BotOrNot: A system to evaluate social bots. February 2016.

[85] Manolis Delakis and Christophe Garcia. text detection with convolutional neural networks. In *VISAPP (2)*, pages 290–294, 2008.

[86] A Demontis, M Melis, B Biggio, D Maiorca, D Arp, K Rieck, I Corona, G Giacinto, and F Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Trans. Dependable Secure Comput.*, pages 1–1, 2018.

[87] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. Pixellink: Detecting scene text via instance segmentation. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[88] Prateek Dewan and Ponnurangam Kumaraguru. Facebook inspector (FbI): Towards automatic real-time detection of malicious content on facebook. *Social Network Analysis and Mining*, 7(1):15, April 2017.

[89] Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. Tweet2vec: Character-based distributed representations for social media. *arXiv preprint arXiv:1605.03481*, 2016.

[90] Dongyang Dou, Jian Jiang, Yuling Wang, and Yong Zhang. A rule-based classifier ensemble for fault diagnosis of rotating machinery. *Journal of Mechanical Science and Technology*, 32(6):2509–2515, 2018.

[91] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[92] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. On adversarial examples for character-level neural machine translation. *arXiv preprint arXiv:1806.09030*, 2018.

[93] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

[94] Juan Echeverría, Emiliano De Cristofaro, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, and Shi Zhou. LOBO – evaluation of generalization deficiencies in twitter bot classifiers. September 2018.

[95] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Compa: Detecting compromised accounts on social networks. In *NDSS*, 2013.

[96] Manuel Egele, Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Towards detecting compromised accounts on social networks. September 2015.

[97] Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. Text processing like humans do: Visually attacking and shielding NLP systems. March 2019.

[98] Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[99] Ibrahim Abu El-Khair. 1.5 billion words arabic corpus. *arXiv preprint arXiv:1611.04033*, 2016.

[100] Nour El-Mawass and Saad Alaboodi. Detecting arabic spammers and content polluters on twitter. In *2016 Sixth International Conference on Digital Information Processing and Communications (ICDIPC)*, pages 53–58. IEEE, 2016.

[101] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Trans. Neural Netw.*, 22(10):1517–1531, October 2011.

[102] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963–2970. IEEE, 2010.

[103] Emilio Ferrara. The history of digital spam. *Communications of the ACM*, 62(8):82–91, 2019.

[104] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Communications of the ACM*, 59(7):96–104, 2016.

[105] Raul S Ferreira, Geraldo Zimbrão, and Leandro GM Alvim. Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency. *Information Sciences*, 488:219–237, 2019.

[106] Lukas Fischer, Lisa Ehrlinger, Verena Geist, Rudolf Ramler, Florian Sobieczky, Werner Zellinger, and Bernhard Moser. Applying ai in practice: key challenges and lessons learned. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 451–471. Springer, 2020.

[107] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[108] Paolo Fornacciari, Monica Mordonini, Agostino Poggi, Laura Sani, and Michele Tomaiuolo. A holistic system for troll detection on twitter. *Computers in Human Behavior*, 89:258–268, 2018.

[109] Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168, 1997.

[110] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44, April 2014.

[111] Hongyu Gao, Yi Yang, Kai Bu, Yan Chen, Doug Downey, Kathy Lee, and Alok Choudhary. Spam ain't as diverse as it seems: throttling OSN spam with templates underneath. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 76–85. ACM, December 2014.

[112] J Gao, W Fan, and J Han. On appropriate assumptions to mine data streams: Analysis and practice. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 143–152, October 2007.

[113] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[114] Zafar Gilani, Reza Farahbakhsh, Gareth Tyson, Liang Wang, and Jon Crowcroft. Of bots and humans (on twitter). In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 349–354, 2017.

[115] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

[116] Ankur Goyal, Likhita Rathore, and Sandeep Kumar. A survey on solution of imbalanced data classification problem using smote and extreme learning machine. In *Communication and Intelligent Systems*, pages 31–44. Springer, 2021.

[117] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[118] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM.

[119] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @ spam: the underground on 140 characters or less. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 27–37, 2010.

[120] Xiaowei Gu and Plamen P Angelov. Highly interpretable hierarchical deep rule-based classifier. *Applied Soft Computing*, page 106310, 2020.

[121] Payas Gupta, Roberto Perdisci, and Mustaque Ahamad. Towards measuring the role of phone numbers in Twitter-Advertised spam. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 285–296. ACM, May 2018.

[122] Srishti Gupta, Dhruv Kuchhal, Payas Gupta, Mustaque Ahamad, Manish Gupta, and Ponnurangam Kumaraguru. Under the shadow of sunshine: Characterizing spam campaigns abusing phone numbers across online social networks. April 2018.

[123] Petr Hajek. Interpretable fuzzy rule-based systems for detecting financial statement fraud. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 425–436. Springer, 2019.

[124] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[125] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[127] Georg Heigold, Günter Neumann, and Josef van Genabith. How robust are character-based word embeddings in tagging and mt against wrod scramlbing or randdm nouse? *arXiv preprint arXiv:1704.04441*, 2017.

[128] Hansi Hettiarachchi and Tharindu Ranasinghe. Emoji powered capsule network to detect type and target of offensive posts in social media. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 474–480, 2019.

[129] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin I P Rubinstein, and J D Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, pages 43–58, New York, NY, USA, 2011. ACM.

[130] Andrew Hutchinson. Facebook updates news feed algorithm to demote misleading health claims. `https://www.socialmediatoday.com/news/facebook-updates-news-feed-algorithm-to-demote-misleading-health-claims/558100/`, July 2019. Accessed: 2019-12-11.

[131] Rodrigo Igawa, Alex Almeida, Bruno Zarpelão, et al. Recognition of compromised accounts on twitter. In *Anais do XI Simpósio Brasileiro de Sistemas de Informação*, pages 9–14. SBC, 2015.

[132] Jane Im, Eshwar Chandrasekharan, Jackson Sargent, Paige Lighthammer, Taylor Denby, Ankit Bhargava, Libby Hemphill, David Jurgens, and Eric Gilbert. Still out there: Modeling and identifying russian troll accounts on twitter. *arXiv preprint arXiv:1901.11162*, 2019.

[133] N. Imam and V. Vassilakis. Detecting spam images with embedded arabic text in twitter. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 6, pages 1–6, Sep. 2019.

[134] NIDDAL IMAM. Health-related spam campaigns, 2020.

[135] NIDDAL IMAM. Health-related spam images, 2021.

[136] NIDDAL IMAM. Synthetic arabic data for scene text recognition, 2021.

[137] NIDDAL IMAM. Synthetic datasets of adversarial imagesn, 2021.

[138] Niddal Imam, Biju Issac, and Seibu Mary Jacob. A semi-supervised learning approach for tackling twitter spam drift. *International Journal of Computational Intelligence and Applications*, 18(02):1950010, 2019.

[139] Niddal Imam and Vassilios Vassilakis. An approach for detecting image spam in OSNs, 2019.

[140] Niddal Imam and Vassilios Vassilakis. Detecting spam images with embedded arabic text in twitter, 2019.

[141] Niddal Imam and Vassilios Vassilakis. Detecting spam images with embedded arabic text in twitter. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 6, pages 1–6. IEEE, 2019.

[142] Niddal H Imam and Vassilios G Vassilakis. A survey of attacks against twitter spam detectors in an adversarial environment. *Robotics*, 8(3):50, 2019.

[143] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. June 2014.

[144] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *Int. J. Comput. Vis.*, 116(1):1–20, January 2016.

[145] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. (1), 2018.

[146] Alex Kantchelian, Sadia Afroz, Ling Huang, Aylin Caliskan Islam, Brad Miller, Michael Carl Tschantz, Rachel Greenstadt, Anthony D Joseph, and J D Tygar. Approaches to adversarial drift. *Proceedings of the 2013 ACM workshop on Artificial intelligence and security - AISec '13*, pages 99–110, 2013.

[147] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.

[148] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1484–1493. IEEE, 2013.

[149] Hamid Karimi, Courtland VanDam, Liyang Ye, and Jiliang Tang. End-to-end compromised account detection. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 314–321. IEEE, 2018.

[150] P Kaur, A Singhal, and J Kaur. Spam detection on twitter: A survey. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2570–2573, March 2016.

[151] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*, 2017.

[152] ukasz Korycki, Alberto Cano, and Bartosz Krawczyk. Active learning with abstaining classifiers for imbalanced drifting data streams. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2334–2343. IEEE, 2019.

[153] ukasz Korycki and Bartosz Krawczyk. Adversarial concept drift detection under poisoning attacks for robust data stream mining. *arXiv preprint arXiv:2009.09497*, 2020.

[154] ukasz Korycki and Bartosz Krawczyk. Online oversampling for sparsely labeled imbalanced and non-stationary data streams. 07 2020.

[155] Kami Kosenko, Emily Winderman, and Abigail Pugh. The hijacked hashtag: The constitutive features of abortion stigma in the# shoutyourabortion twitter campaign. *International Journal of Communication*, 13:21, 2019.

[156] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In *Fifth International AAAI conference on weblogs and social media*, 2011.

[157] Bartosz Krawczyk and Michał Woźniak. Online query by committee for active learning from drifting data streams. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2120–2127. IEEE, 2017.

[158] Sanjay Krishnan and Eugene Wu. Palm: Machine learning explanations for iterative debugging. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, pages 1–6, 2017.

[159] Paweł Ksieniewicz, Michał Woźniak, Bogusław Cyganek, Andrzej Kasprzak, and Krzysztof Walkowiak. Data stream classification using active learned neural networks. *Neurocomputing*, 353:74–82, 2019.

[160] Peter Kubat. Assessing reliability of modular software. *Operations research letters*, 8(1):35–41, 1989.

[161] Ludmila I Kuncheva and Combining Pattern Classifiers. Methods and algorithms. *Jonh Wiley & Sons, New York, NY*, 2004.

[162] Keita Kurita, Anna Belova, and Antonios Anastasopoulos. Towards robust toxic content classification. *arXiv preprint arXiv:1912.06872*, 2019.

[163] Ricky Laishram and Vir Virander Phoha. Curie: A method for protecting svm classifier from poisoning attack. *arXiv preprint arXiv:1606.01584*, 2016.

[164] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

[165] L A Lalitha, V R Hulipalled, and K R Venugopal. Spamming the mainstream: A survey on trending twitter spam detection techniques. In *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, pages 444–448, August 2017.

[166] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[167] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[168] Yann LeCun, Yoshua Bengio, and Others. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[169] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[170] Sangho Lee and Jong Kim. Early filtering of ephemeral malicious accounts on twitter. *Computer Communications*, 54:48–57, 2014.

[171] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[172] Yue Li, Pengjian Xu, and Minmin Pang. Adversarial attacks on word2vec and neural network. In *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, page 50. ACM, December 2018.

[173] M Liao, B Shi, and X Bai. TextBoxes++: A Single-Shot oriented scene text detector. *IEEE Trans. Image Process.*, 27(8):3676–3690, August 2018.

[174] Minghui Liao, Baoguang Shi, Xiang Bai, Xinggang Wang, and Wenyu Liu. Textboxes: A fast text detector with a single deep neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[175] G Lin, N Sun, S Nepal, J Zhang, Y Xiang, and H Hassan. Statistical twitter spam detection demystified: Performance, stability and scalability. *IEEE Access*, 5:11142–11154, 2017.

[176] Guanjun Lin, Nan Sun, Surya Nepal, Jun Zhang, Yang Xiang, and Houcine Hassan. Statistical twitter spam detection demystified: performance, stability and scalability. *IEEE access*, 5:11142–11154, 2017.

[177] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[178] Shigang Liu, Jun Zhang, and Yang Xiang. Statistical detection of online drifting twitter spam: Invited paper. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 1–10. ACM, May 2016.

[179] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[180] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. Textsnake: A flexible representation for detecting text of arbitrary shapes. In *ECCV*, 2018.

[181] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

[182] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 641–647, 2005.

[183] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *CEAS*, volume 2005, 2005.

[184] Daniel Lowd and Christopher Meek. Good Word Attacks on Statistical Spam Filters. *Conference on email and anti-spam*, 2005.

[185] Sreekanth Madisetty and Maunendra Sankar Desarkar. A neural network-based ensemble approach for spam detection in twitter. *IEEE Transactions on Computational Social Systems*, 5(4):973–984, 2018.

[186] Osama A Mahdi, Eric Pardede, Nawfal Ali, and Jinli Cao. Fast reaction to sudden concept drift in the absence of class labels. *Applied Sciences*, 10(2):606, 2020.

[187] Sebastián Maldonado, Julio López, and Carla Vairetti. An alternative smote oversampling strategy for high-dimensional datasets. *Applied Soft Computing*, 76:380–389, 2019.

[188] Ofer Matan. On voting ensembles of classifiers. In *Proceedings of AAAI-96 workshop on integrating multiple learned models*, pages 84–88. Citeseer, 1996.

[189] M Mataoui, O Zelmati, D Boughaci, M Chaouche, and F Lagoug. A proposed spam detection approach for arabic social networks content. In *2017 International Conference on Mathematics and Information Technology (ICMIT)*, pages 222–226, December 2017.

[190] M Mateen, M A Iqbal, M Aleem, and M A Islam. A hybrid approach for spam detection for twitter. In *2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 466–471, January 2017.

[191] Michele Mazza, Stefano Cresci, Marco Avvenuti, Walter Quattrociocchi, and Maurizio Tesconi. Rtbust: exploiting temporal patterns for botnet detection on twitter. In *Proceedings of the 10th ACM Conference on Web Science*, pages 183–192, 2019.

[192] C Meda, E Ragusa, C Gianoglio, R Zunino, A Ottaviano, E Scillia, and R Surlinelli. Spam detection of twitter traffic: A framework based on random forests and non-uniform feature sampling. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 811–817, August 2016.

[193] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. May 2018.

[194] Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. Finding opinion manipulation trolls in news community forums. In *Proceedings of the nineteenth conference on computational natural language learning*, pages 310–314, 2015.

[195] Brad Miller, Alex Kantchelian, Sadia Afroz, Rekha Bachwani, Edwin Dauber, Ling Huang, Michael Carl Tschantz, Anthony D Joseph, and J D Tygar. Adversarial Active Learning. *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop - AISec '14*, (August):3–14, 2014.

[196] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, and Alex Hai Wang. Twitter spammer detection using data stream clustering. *Information Sciences*, 260:64–73, 2014.

[197] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

[198] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[199] Noam Mor and Lior Wolf. Confidence prediction for lexicon-free ocr. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 218–225. IEEE, 2018.

[200] Fred Morstatter, Liang Wu, Tahora H Nazer, Kathleen M Carley, and Huan Liu. A new approach to bot detection: striking the balance between precision and recall. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 533–540. IEEE, 2016.

[201] Busra Mutlu, Merve Mutlu, Kasım Oztoprak, and Erdogan Dogdu. Identifying trolls and determining terror awareness level in social networks using a scalable framework. In *2016 IEEE international conference on big data (big data)*, pages 1792–1798. IEEE, 2016.

[202] Meike Nauta. Detecting hacked twitter accounts by examining behavioural change using twtter metadata. In *Proceedings of the 25th Twente Student Conference on IT*, 2016.

[203] N Nayef, F Yin, I Bizid, H Choi, Y Feng, D Karatzas, Z Luo, U Pal, C Rigaud, J Chazalon, W Khlif, M M Luqman, J Burie, C Liu, and J Ogier. ICDAR2017 robust reading challenge on Multi-Lingual scene text detection and script identification - RRC-MLT. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1454–1459, November 2017.

[204] Lukas Neumann and Jiri Matas. A method for text localization and recognition in real-world images. In *Asian conference on computer vision*, pages 770–783. Springer, 2010.

[205] James Newsome, Brad Karp, and Dawn Song. Paragraph: Thwarting signature learning by training maliciously. In *Lecture Notes in Computer Science*, pages 81–105. 2006.

[206] Shirin Nilizadeh, Francois Labrèche, Alireza Sedighian, Ali Zand, José Fernandez, Christopher Kruegel, Gianluca Stringhini, and Giovanni Vigna. POISED: Spotting twitter spam off the beaten paths. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1159–1174. ACM, October 2017.

[207] Peter Norvig. How to write a spelling corrector. *Online at: http://norvig. com/spell-correct. html*, 2007.

[208] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[209] Firoj Parwej. An empirical evaluation of off-line arabic handwriting and printed characters recognition system. *International Journal of Computer Science Issues (IJCSI)*, 9(6):29, 2012.

[210] Aryo Pinandito, Rizal Setya Perdana, Mochamad Chandra Saputra, and Hanifah Muslimah Az-zahra. Spam detection framework for android twitter application using naïve bayes and K-Nearest neighbor classifiers. In *Proceedings of the 6th International Conference on Software and Computer Applications*, pages 77–82. ACM, February 2017.

[211] Alberto Poncelas, Mohammad Aboomar, Jan Buts, James Hadley, and Andy Way. A tool for facilitating ocr postediting in historical documents. *arXiv preprint arXiv:2004.11471*, 2020.

[212] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[213] Christophe Rigaud, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. Icdar 2019 competition on post-ocr text correction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1588–1593. IEEE, 2019.

[214] Sergio Rojas-Galeano. On obstructing obscenity obfuscation. *ACM Transactions on the Web (TWEB)*, 11(2):1–24, 2017.

[215] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[216] Radwa MK Saeed, Sherine Rady, and Tarek F Gharib. An ensemble approach for spam detection in arabic opinion texts. *Journal of King Saud University-Computer and Information Sciences*, 2019.

[217] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

[218] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. July 2017.

[219] Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. Humpty dumpty: Controlling word meanings via corpus poisoning. *arXiv preprint arXiv:2001.04935*, 2020.

[220] D Sculley, Matthew Eric Otey, Michael Pohl, Bridget Spitznagel, John Hainsworth, and Yunkai Zhou. Detecting adversarial advertisements in the wild. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 274–282. ACM, 2011.

[221] Surendra Sedhai and Aixin Sun. HSpam14: A collection of 14 million tweets for Hashtag-Oriented spam research. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 223–232. ACM, August 2015.

[222] Surendra Sedhai and Aixin Sun. Semi-supervised spam detection in twitter stream. *IEEE Transactions on Computational Social Systems*, 5(1):169–175, 2017.

[223] Tegjyot Singh Sethi. Dynamic adversarial mining-effectively applying machine learning in adversarial non-stationary environments. 2017.

[224] Tegjyot Singh Sethi and Mehmed Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. March 2017.

[225] Tegjyot Singh Sethi and Mehmed Kantardzic. Data driven exploratory attacks on black box classifiers in adversarial domains. *Neurocomputing*, 289:129–143, 2018.

[226] Tegjyot Singh Sethi and Mehmed Kantardzic. Handling adversarial concept drift in streaming data. *Expert Syst. Appl.*, 97:18–40, 2018.

[227] Tegjyot Singh Sethi and Mehmed Kantardzic. Handling adversarial concept drift in streaming data. *Expert Syst. Appl.*, 97:18–40, 2018.

[228] Tegjyot Singh Sethi and Mehmed Kantardzic. Handling adversarial concept drift in streaming data. *Expert Syst. Appl.*, 97:18–40, 2018.

[229] Tegjyot Singh Sethi and Mehmed Kantardzic. Handling adversarial concept drift in streaming data. *Expert Syst. Appl.*, 97:18–40, 2018.

[230] Tegjyot Singh Sethi and Mehmed Kantardzic. Handling adversarial concept drift in streaming data. *Expert systems with applications*, 97:18–40, 2018.

[231] Tegjyot Singh Sethi, Mehmed Kantardzic, Lingyu Lyu, and Jiashun Chen. A dynamic-adversarial mining approach to the security of machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(3):e1245, 2018.

[232] Dominic Seyler, Lunan Li, and Chengxiang Zhai. Identifying compromised accounts on social media using statistical text analysis. April 2018.

[233] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting oriented text in natural images by linking segments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2550–2558, 2017.

[234] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[235] Baoguang Shi, Xiang Bai, and Cong Yao. An End-to-End trainable neural network for Image-Based sequence recognition and its application to scene text recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(11):2298–2304, November 2017.

[236] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4168–4176, 2016.

[237] Rosa Sicilia, Stella Lo Giudice, Yulong Pei, Mykola Pechenizkiy, and Paolo Soda. Twitter rumour detection in the health domain. *Expert Syst. Appl.*, 110:33–40, November 2018.

[238] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[239] Cooper Smith. Facebook users are uploading 350 million new photos each day. *Business insider*, 18, 2013.

[240] Congzheng Song and Vitaly Shmatikov. Fooling ocr systems with adversarial text images. *arXiv preprint arXiv:1802.05385*, 2018.

[241] Congzheng Song and Vitaly Shmatikov. Fooling OCR systems with adversarial text images. February 2018.

[242] Jonghyuk Song, Sangho Lee, and Jong Kim. CrowdTarget. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, (i):793–804, 2015.

[243] Jonghyuk Song, Sangho Lee, and Jong Kim. CrowdTarget. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, (i):793–804, 2015.

[244] Charles Spearman. The proof and measurement of association between two things. 1961.

[245] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 1–9, New York, NY, USA, 2010. ACM.

[246] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[247] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[248] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. December 2013.

[249] Kazem Taghva and Eric Stofsky. Ocrspell: an interactive spelling correction system for ocr errors in text. *International Journal on Document Analysis and Recognition*, 3(3):125–137, 2001.

[250] Kurt Thomas, Frank Li, Chris Grier, and Vern Paxson. Consequences of connectivity: Characterizing account hijacking on twitter. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 489–500. ACM, November 2014.

[251] Paul Thompson, John McNaught, and Sophia Ananiadou. Customised ocr correction for historical medical text. In *2015 Digital Heritage*, volume 1, pages 35–42. IEEE, 2015.

[252] Yingjie Tian and Saiji Fu. A descriptive framework for the field of deep learning applications in medical images. *Knowledge-Based Systems*, 210:106445, 2020.

[253] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision*, pages 56–72. Springer, 2016.

[254] Maroua Tounsi, Ikram Moalla, Adel M Alimi, and Frank Lebouregois. Arabic characters recognition in natural scenes using sparse coding for feature representations. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1036–1040. IEEE, 2015.

[255] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2005–2021, 2019.

[256] D Trång, F Johansson, and M Rosell. Evaluating algorithms for detection of compromised social media user accounts. In *2015 Second European Network Intelligence Conference*, pages 75–82, September 2015.

[257] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[258] Courtland VanDam, Farzan Masrour, Pang-Ning Tan, and Tyler Wilson. You have been caute! early detection of compromised accounts on social media. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 25–32, 2019.

[259] Courtland VanDam, Pang-Ning Tan, Jiliang Tang, and Hamid Karimi. Cadet: A multi-view learning framework for compromised account detection on twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 471–478. IEEE, 2018.

[260] Courtland VanDam, Pang-Ning Tan, Jiliang Tang, and Hamid Karimi. CADET: A Multi-View learning framework for compromised account detection on twitter, 2018.

[261] Courtland VanDam, Jiliang Tang, and Pang-Ning Tan. Understanding compromised accounts on twitter. In *Proceedings of the International Conference on Web Intelligence*, pages 737–744, 2017.

[262] Onur Varol, Emilio Ferrara, Clayton Davis, Filippo Menczer, and Alessandro Flammini. Online human-bot interactions: Detection, estimation, and characterization. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11, 2017.

[263] Onur Varol, Emilio Ferrara, Clayton A Davis, Filippo Menczer, and Alessandro Flammini. Online Human-Bot interactions: Detection, estimation, and characterization. March 2017.

[264] Prashanth Vijayaraghavan and Deb Roy. Generating black-box adversarial examples for text classifiers using a deep reinforced model. *arXiv preprint arXiv:1909.07873*, 2019.

[265] Bimal Viswanath, M Ahmad Bashir, Mark Crovella, Saikat Guha, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Towards detecting anomalous user behavior in online social networks. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 223–238, 2014.

[266] Martin Volk, Lenz Furrer, and Rico Sennrich. Strategies for reducing and correcting ocr errors. In *Language technology for cultural heritage*, pages 3–22. Springer, 2011.

[267] Boxin Wang, Hengzhi Pei, Han Liu, and Bo Li. Advcodec: Towards a unified framework for adversarial text generation. *arXiv preprint arXiv:1912.10375*, 2019.

[268] De Wang, Shamkant Navathe, Ling Liu, Danesh Irani, Acar Tamersoy, and Calton Pu. Click traffic analysis of short URL spam on twitter. ICST, November 2013.

[269] Gang Wang, Santa Barbara, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs . Machine : Practical Adversarial Detection of Malicious Crowdsourcing Workers. *the 23rd USENIX Security Symposium*, pages 239–254, 2014.

[270] Gang Wang, Santa Barbara, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs . Machine : Practical Adversarial Detection of Malicious Crowdsourcing Workers. *the 23rd USENIX Security Symposium*, pages 239–254, 2014.

[271] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 239–254, 2014.

[272] Tinghua Wang, Lin Zhang, and Wenyu Hu. Bridging deep and multiple kernel learning: A review. *Information Fusion*, 2020.

[273] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

[274] Mahdi Washha, Aziz Qaroush, and Florence Sedes. Leveraging time for spammers detection on twitter. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems*, MEDES, pages 109–116, New York, NY, USA, 2016. ACM.

[275] Tingmin Wu, Shigang Liu, Jun Zhang, and Yang Xiang. Twitter spam detection based on deep learning. In *Proceedings of the australasian computer science week multiconference*, pages 1–8, 2017.

[276] Tingmin Wu, Sheng Wen, Yang Xiang, and Wanlei Zhou. Twitter spam detection: Survey of new approaches and comparative study. *Comput. Secur.*, 76:265–284, July 2018.

[277] C Yang, R Harkreader, and G Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Trans. Inf. Forensics Secur.*, 8(8):1280–1293, August 2013.

[278] Chao Yang, Robert Chandler Harkreader, and Guofei Gu. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers, 2011.

[279] Huanrui Yang, Jingyang Zhang, Hongliang Dong, Nathan Inkawhich, Andrew Gardner, Andrew Touchet, Wesley Wilkes, Heath Berry, and Hai Li. Dverge: Diversifying vulnerabilities for enhanced robust generation of ensembles. *arXiv preprint arXiv:2009.14720*, 2020.

[280] Wei Yang, Deguang Kong, Tao Xie, and Carl A Gunter. Malware Detection in Adversarial Settings. *Proceedings of the 33rd Annual Computer Security Applications Conference on - ACSAC 2017*, pages 288–302, 2017.

[281] Xuebing Yang, Qiuming Kuang, Wensheng Zhang, and Guoping Zhang. Amdo: An over-sampling technique for multi-class imbalanced problems. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1672–1685, 2017.

[282] Kan Yuan, Di Tang, Xiaojing Liao, Xiaofeng Wang, Xuan Feng, Yi Chen, Menghan Sun, Haoran Lu, and Kehuan Zhang. Stealthy porn: Understanding Real-World adversarial images for illicit online promotion, 2019.

[283] Kan Yuan, Di Tang, Xiaojing Liao, XiaoFeng Wang, Xuan Feng, Yi Chen, Menghan Sun, Haoran Lu, and Kehuan Zhang. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *Stealthy Porn: Understanding Real-World Adversarial Images for Illicit Online Promotion*, page 0. IEEE, 2019.

[284] Oussama Zayene, Jean Hennebert, Sameh Masmoudi Touj, Rolf Ingold, and Najoua Essoukri Ben Amara. A dataset for arabic text detection, tracking and recognition in news videos-activ. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 996–1000. IEEE, 2015.

[285] Oussama Zayene, Sameh Masmoudi Touj, Jean Hennebert, Rolf Ingold, and Najoua Essoukri Ben Amara. Multi-dimensional long short-term memory networks for artificial arabic text recognition in news video. *IET Computer Vision*, 12(5):710–719, 2018.

[286] Mingming Zha, Guozhu Meng, Chaoyang Lin, Zhe Zhou, and Kai Chen. Rolma: A practical adversarial attack against deep learning-based lpr systems. In *International Conference on Information Security and Cryptology*, pages 101–117. Springer, 2019.

[287] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

[288] X Zhang, S Zhu, and W Liang. Detecting spam and promoting campaigns in the twitter social network. In *2012 IEEE 12th International Conference on Data Mining*, pages 1194–1199, December 2012.

[289] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.

[290] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.

[291] T Zhu, H Gao, Y Yang, K Bu, Y Chen, D Downey, K Lee, and A N Choudhary. Beating the artificial chaos: Fighting OSN spam using its own templates. *IEEE/ACM Trans. Netw.*, 24(6):3856–3869, December 2016.

[292] Tuanfei Zhu, Yaping Lin, and Yonghe Liu. Synthetic minority oversampling technique for multiclass imbalance problems. *Pattern Recognition*, 72:327–340, 2017.

[293] Yingying Zhu, Cong Yao, and Xiang Bai. Scene text detection and recognition: recent advances and future trends. *Frontiers of Computer Science*, 10(1):19–36, February 2016.

[294] Guido Zuccon, Anthony N Nguyen, Anton Bergheim, Sandra Wickman, and Narelle Grayson. The impact of ocr accuracy on automated cancer classification of pathology reports. In *HIC*, pages 250–256, 2012.