# Detection of Software Vulnerability Communication in Expert Social Media Channels: A Data-driven Approach

Andrei Lima Queiroz
*Technological University Dublin*

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH

DUBLIN

TECHNOLOGICAL
UNIVERSITY DUBLIN

# Detection of Software Vulnerability Communication in Expert Social Media Channels: A Data-driven Approach

by

**Andrei Lima Queiroz**

Supervisors: Dr. Brian Keegan
Dr. Susan McKeever

School of Computer Science

A thesis submitted to the Technological University Dublin for the degree of Doctor of Philosophy

September, 2020

# Declaration

I certify that this thesis which I now submit for examination for the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for graduate study by research of the Technological University Dublin and has not been submitted in whole or in part for another award in any other third level institution.

The work reported on in this thesis conforms to the principles and requirements of the TU Dublin's guidelines for ethics in research. TU Dublin has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Signed: ____________________ Date: ____________________

# Abstract

Conceptually, a vulnerability is: "*A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy*". Some of these flaws can go undetected and exploited for long periods of time after software release. Although some software providers are making efforts to avoid this situation, inevitability, users are still exposed to vulnerabilities that allow criminal hackers to take advantage. These vulnerabilities are constantly discussed in specialised forums on social media. Therefore, from a cyber security standpoint, the information found in these places can be used for countermeasures actions against malicious exploitation of software. However, manual inspection of the vast quantity of shared content in social media is impractical. For this reason, in this thesis, we analyse the real applicability of supervised classification models to automatically detect software vulnerability communication in expert social media channels. We cover the following three principal aspects:

Firstly, we investigate the applicability of classification models in a range of 5 different datasets collected from 3 Internet Domains: Dark Web, Deep Web and Surface Web. Since supervised models require labelled data, we have provided a systematic labelling process using multiple annotators to guarantee accurate labels to carry out experiments. Using these datasets, we have investigated the classification models with different combinations of learning-based algorithms and traditional features representation. Also, by oversampling the positive instances, we have achieved an increase of 5% in Positive Recall (on average) in these models. On top of that, we have applied Feature Reduction, Feature Extraction and Feature Selection techniques, which provided a reduction on the dimensionality of these models without damaging the accuracy, thus, providing computationally efficient models.

Furthermore, in addition to traditional features representation, we have investigated the performance of robust language models, such as Word Embedding (WEMB) and Sentence Embedding (SEMB) on the accuracy of classification models. Regarding WEMB, our experiment has shown that this model trained with a small security-vocabulary dataset provides comparable results with WEMB trained in a very large general-vocabulary dataset. Regarding SEMB model, our experiment has shown that its use overcomes WEMB model in detecting vulnerability communication, recording 8% of Avg. Class Accuracy and 74% of Positive Recall. In addition, we investigate two Deep Learning algorithms as classifiers, text CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network)-based algorithms, which have improved our model, resulting in the best overall performance for our task.

Finally, we simulate the deployment of these models in a real-life situation for 1-year period. The results suggest that Concept Drift affects the performance of models created with hacker forums data (Deep Web), which has presented a faster performance degradation than models created with Twitter data (Surface Web). Additionally, to avoid a sudden decrease in performance and provide more robust models over time, we presented the optimal proportion of 50% re-train, which reduce in half the human workload needed for performing the labelling tasks; and also presented that the weight proportion of 3:1 on the newest instances enhance the performance of the model.

# Acknowledgements

The life of a Ph.D. student seems a lonely journey, but when I look back, I see how many people have supported me with this challenge.

Firstly, I would like to thank my supervisors, Dr. Brian Keegan and Dr. Susan McKeever. Brian, thank you for your friendship, encouragement and thoughtful advice, which have helped to grow as a researcher and professional. Susan, thank you for joining the team, this research would not have taken off the way it did without your expertise and vision, you know that.

I would like to thank my wife, Daniela Garcia, who has joined me in this adventure called "Ph.D. in a foreign country". Thanks for supporting me and for being by my side in all moments. Without you, I wouldn't have had the peace of mind and the necessary comfort to get through the harsh days.

Also, I would like to thank my nuclear family composed by my mother, Francisca, to whom I owe all my achievements. Thank you for your unconditional belief, support and love. Also, my brother, Cleiton and stepfather, José, for supporting me and for taking care of the family during my sojourning abroad.

I am grateful to Dr. J. Paul Gibson and Dr. Svetlana Hensman for accepting to be my examiners and for sharing their expertise with me during the viva.

I would like to thank some brilliant Ph.D. students, whom I had a pleasure to meet in TU Dublin: Hao, Husam, Giuliano, Giancarlo, Pierre, Fei, Pallavi, Jack, Ivan, Xuehao, Annika, Mariana, Lucas, Vihanga, Senja. Sharing the same place with these fellows has made me learn a lot.

I cannot forget to thank some of TU Dublin lectures who, in some way, have inspired me with their commitment to the academy and science. A special thanks to Prof. Sarah Jane Delany, who was always willing to help me, Dr. Luca Longo, and Dr. Pierpaolo Dondio.

I would like to thank my friends Victor and Slawek for helping out with their wisdom in the earliest phase of data collection. Also, I cannot forget Dr. Fredrick Mtenzi for opening the doors of the former Dublin Institute of Technology (DIT) for me.

I am very grateful to the Universidade de Brasília (UnB), which has provided me with the opportunity of diving into the sea of knowledge from academic and professional perspectives. A special thanks to Prof. Marcelo Ladeira, Prof. Jorge Fernandes, and Prof. Jacir Bordim, who have been promoting employee career progression through academic achievements. And to STI Analysts Riane, Karam, Domingos, and Jackson for helping me with this goal.

Finally, I would like to thank CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for my scholarship under the Science Without Borders programme, number 201898/2015-2.

Go raiḃ míle maiṫ agaiḃ go léir

# Contents

# List of Figures

# List of Tables

# CHAPTER 1

# Introduction

There is no guarantee that we are using software products free of vulnerabilities. According to Shirey (2007), vulnerability is "A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy". Some of these flaws are built-in to software products and can remain unknown or dormant for long periods. Elements of these flaws are inserted by mistake during the development of software due to the lack of training on security development and quality assurance procedures (Acar et al., 2017, Younis et al., 2016). Thus, leaving a path for exposure of information of users to hackers with malicious intentions.

According to an important source of information about software vulnerabilities, the National Vulnerability Database (NVD), the number of security issues have been increasing each year, with a disclosing of 4,000 vulnerabilities in 2010 up to 17,000 in 2019 (NIST, 2019). As seen in Figure 1.1, there is an ascendant trend line with a sudden increase in the number of vulnerable software numbers in 2017, 2018 and 2019.

This ascendant trend might be related to the growing number of services offered by computer-like devices which modern society is increasingly reliant on, such as the proliferation of Internet of Things (IoT). We, as software users, are using more software

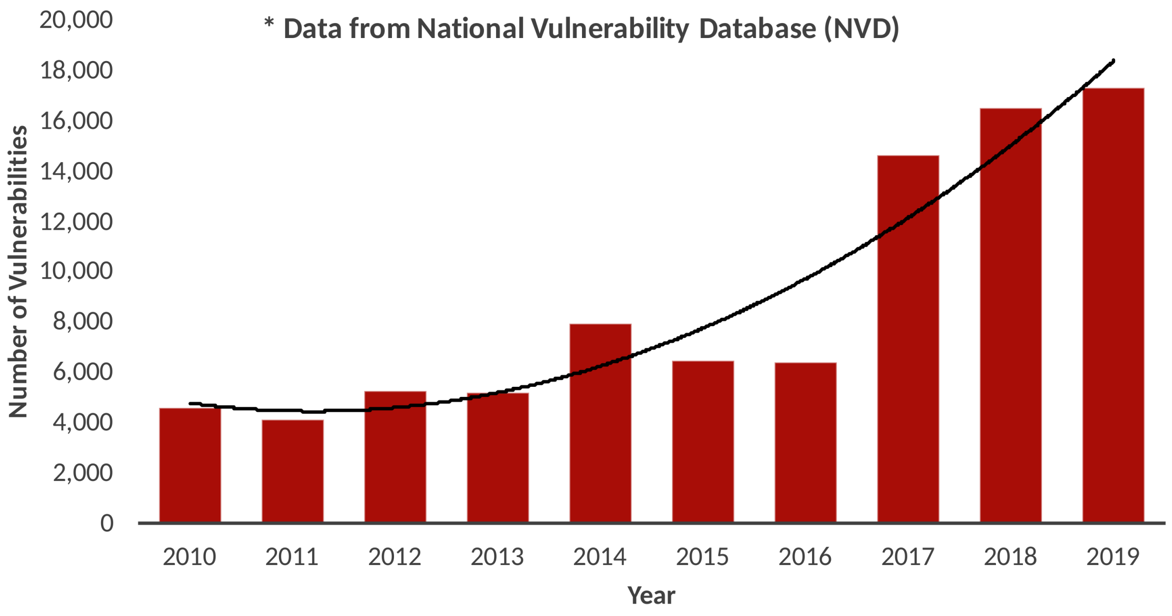


Figure 1.1: The Growth of Software Vulnerabilities in the Last 10 years (NIST, 2019)

than ever before. Once, it was used only within strict boundaries, such as our desktop computer and home network: nowadays, it is ubiquitous and connected to other services around the world. Software is present in several aspects of our life, for instance, shopping, entertainment, career, etc. Since the popularisation of the Internet, there has been an increasing demand for computer-like devices that are not only restricted to our desk computers, instead, they are found in mobile phones in our pockets, home appliances (IoT), cars, and nowadays these technologies can be used for monitoring entire cities (Smart Cities).

However, the nature of these devices has created a wide range of potential attack vectors. According to Jang-Jaccard and Nepal (2014), embedded systems and sensors, which are part of the IoT paradigm, are the topics that have been receiving the most attention from industry and academia in recent years due to their increased use in our lives. Such devices lack proper maintenance and security updating (patching), which leave their vulnerabilities exposed to hacker attacks.

It has been frequently reported that hackers explore software vulnerabilities to exfiltrate data (Do et al., 2016, D'Orazio et al., 2017, Thompson et al., 2016). Such attacks not only affect a massive number of users but also incur a financial loss and damage to a company's reputation. As an example, we see the data breach attack on the Equifax credit company, which affected the private information of more than 140 million people. This breach originated from a vulnerability on Equifax applications and it is believed that the data was shared (or sold) on Dark Web underground forums (CNBC, 2019). Another example occurred on the Facebook web application and affected 50 million users. This breach occurred through a flaw found in the "View As" feature, which allowed a hacker to exfiltrate users authorization token. After that event, Facebook's security team acknowledged that the vulnerability was introduced in the application around July 2017 and remained hidden for more than one-year (Facebook Newsroom, 2018).

Both incidents occurred through security flaws in their software application, although the principal difference is that, in the Facebook case, the vulnerability was explored before the release of a security patch (black area), whereas in Equifax it was explored after that event (grey area) as seen in Figure 1.2. Moreover, it is worth highlight that the insecure version of the software remained publicly available with a hidden vulnerability for a long time before discovery, 303 days for Equifax, and 453 days for Facebook. According to Bilge and Dumitraş (2012), hidden vulnerabilities might be exploited by a few experts, however, after its public disclosure, the volume of exploitation can go up to 5 orders of magnitude. This is what happened in Equifax case, the attack occurred in a time window after public disclosure and before the update patch being applied (144 days of time window), whereas, for Facebook, the attack occurred before public disclosure.

Although the magnitude of the mentioned cases was well explored by specialised media, these were not the only vulnerabilities exploited by hackers. In the following, we present some examples of known critical security flaws that affected (or had a huge potential of affecting) software users, business and governmental institutions:

- During the COVID-19 pandemic, when hundreds of people were lockdown in their homes, the use of video-conference increased suddenly, and Zoom soft-

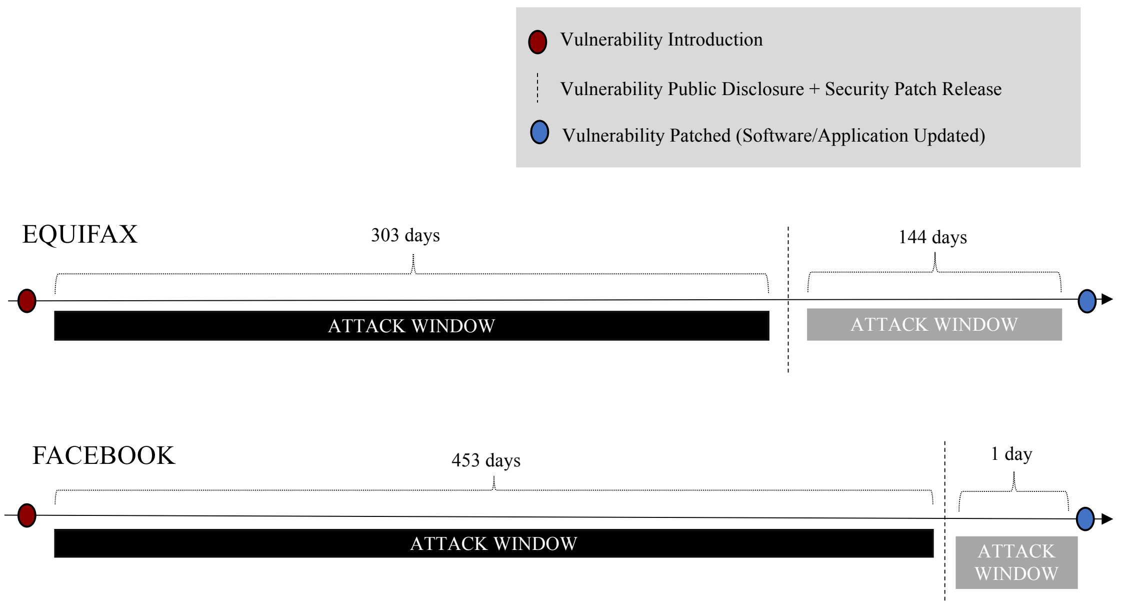

Vulnerability Introduction
Vulnerability Public Disclosure + Security Patch Release
Vulnerability Patched (Software/Application Updated)
EQUIFAX
303 days
144 days
ATTACK WINDOW
ATTACK WINDOW
FACEBOOK
453 days
1 day
ATTACK WINDOW
ATTACK WINDOW

Figure 1.2: Comparison of Attack Windows Between Facebook and Equifax Vulnerability

ware faced this huge increase by going from 10 to 300 million daily meetings in March and April (CNET, 2020a). In this period, several vulnerabilities in Zoom were discovered, which included flaws on Windows and Mac OS software the allowed personal data to be exfiltrated (CNET, 2020b), which, according to Bleeping Computer (2020), was found being sold on Dark Web markets. However, the most notable of these vulnerabilities was named **Zoombombing**, a flaw on the function that generates the Meeting ID, which allows the attacker to discover it by randomly searching valid digits within a space of 9 to 11 digits (krebsonsecurity.com, 2020). Using the flaw, hackers have invaded and disturbed several meetings and webinars around the world.

- The **Wannacry Ransomware**, built to take advantage of vulnerability in Windows Server Message Block (SMB) protocol, affected several companies around the world that had not applied the Microsoft Security Update (MS17-010). This malware, when installed in the host machine, encrypts the files in the hard drive, making it impossible to use. After the encryption, the malware requires a payment (ransom) to provide a key that decrypts the files (Chen and Bridges, 2017). More than 150 countries were affected by this malware in 2017. This has disrupted several services, including the British National Health (NHS)(Ghafur et al., 2019), with a total revenue loss of 92 million pounds (ZDnet, 2018). It also has stopped airline and bank services throughout Europe (theverge.com, 2017).

- In March 2012, the version 1.0.1 of OpenSSL, a widely used cryptographic tool, was released with a vulnerability that allowed the attackers to read sensitive information from a server's memory, including private data, cryptographic keys, login credentials. This vulnerability, named as **Heartbleed** and assigned as CVE-2014-0160 by NVD database, remained hidden (publicly unknown) for 2 years until a patch was provided in April 2014. Researchers considered this vulnerability one of the most impactful flaws found in software due to the widespread use of HTTPS servers around the world (Durumeric et al., 2014). Nobody knows whether this vulnerability was exploited by a hacker during this time, although there are some incidents which are associated with this flaw. For instance, incidents with Canadian Governmental Information systems are associated with Heartbleed vulnerability, including a major data breach on Canadian Revenue Agency (CBC.ca, 2015).
- **Spectre and Meltdown** are exploits created to take advantage of a hardware-level vulnerability found in Intel x86, AMD and ARM-based processors. The Spectre exploit takes advantage of a processor's routine, called *speculative branch*. This routine is manipulated by the exploit to run a set of instructions which should not have been executed under correct program execution (Kocher et al., 2019). Whereas Meltdown takes advantage of *out-of-order execution* processors' execution routine to leak information from kernel-space memory (Lipp et al., 2018). Both attacks can be used to leak private data, for instance, passwords, cryptography keys and certificates. Two main problems make hardware-level vulnerabilities more difficult to patch than software-level vulnerabilities: (1) the patches come with a computational performance cost, compromising a maxi-

mum of 74% of CPU speed (Kocher et al., 2019), and (2) the vulnerabilities might be found in large part of hardwares that use CPUs from the mentioned companies (desktop and computer servers). This is due to the size of market Intel and AMD share in manufacturing and selling CPUs. Recently, new CPUs are being produced with a built-in fix, however, these vulnerabilities will exist in legacy devices for long periods, as consumers need to purchase a new generation of processors to fix this problem permanently.

According to Bilge and Dumitraş (2012), there is no study about the duration and prevalence of attacks towards hidden vulnerabilities before its public disclosure. Also, Rescorla (2005) contribute with the idea that it is hard to determine quantitatively how often attacks to publicly unknown vulnerabilities take place. Moreover, these problems contributed to a debate on whether software vendors are liable for any eventual loss users might have by eventual damage caused by vulnerabilities in software Rice (2007).

Other studies have shown that exploitation of software vulnerabilities can be discovered by tracking specific-purpose social media channels, such as hacker forums, marketplaces and microblogs (Khandpur et al., 2017, Nunes et al., 2016, Sabottke et al., 2015). These social media channels are being used as an ecosystem to share and gain knowledge about hacking techniques, tools and exploitable vulnerabilities in which the principal motivation is mainly financial (Ablon et al., 2014, Roumani et al., 2016). Moreover, Macdonald et al. (2015) reported that hackers eventually use Deep Web forums to exchange security information regarding attacks against critical infrastructures computers, for instance, power grids, financial institution and transportations networks, where communication about such attacks was discovered before the attacks occurred. Additionally, Horawalavithana et al. (2019) reported that Surface Web social media such as Twitter, Reddit and Github that are also being used for security discussion regarding vulnerabilities problems.

In this context, cyber security initiatives are motivated to act proactively against such threats by focusing on which is called Open Source Intelligence (OSINT). OSINT is "data collected from publicly available sources, such as social media, internet and public data, to be used in an intelligence context" (Steele, 1996). To acquire insights from these unstructured data, companies are investing more in data-driven methods that use machine learning algorithms and natural language processing (LLC, 2017). These approaches are experiencing considerable growth due to their capacity for solving real-world problems through the discovery of useful patterns in data. These methods have revolutionised tasks such as image recognition (e.g., handwriting and face classification) and natural language understanding (e.g., sentiment analysis, Named-entity recognition).

In this applied research, we provide a thorough study of these methods with the purpose of detecting vulnerability communications in hacker social media channels. We perform experiments using data-driven approaches across multiple datasets from different social media sources (Surface Web, Deep Web, and Dark Web). We believe that the outcome of our research contributes to cyber security initiatives that aim to use open source information proactively against malicious exploitation of software vulnerabilities.

## 1.1 Background

Monetary gains have been promoting the interest for software vulnerabilities and its trade on underground markets. These vulnerabilities are generally exploited through specific tools (exploits). The market value of these exploits is given by criticality (damage it might cause) and novelty of these vulnerabilities (zero-day vulnerabilities) (Huang et al., 2016). These products are frequently offered on hidden hacker channels on social media, such as Dark Web forums and marketplaces. According to Algarni and Malaiya (2014), underground hacker forums offer more financial advantage to the exploit's seller than legal bounty initiatives, for instance, BugCrowd (www.bugcrowd.com) and HackerOne (www.hackerone.com). Additionally, these forums also work as learning ecosystems, where hackers interact with peers to acquire knowledge on security and technical information.

However, on the other side of the vulnerability ecosystem, we found the "white hats". They are usually security researchers who also consume information from specialised social media channels intending to protect systems against malicious attacks. Moreover, they are also an important part of the *Coordinate Disclosure System* (Pupillo et al., 2018, Shepherd, 2013). This system provides a coordinate agreement between the person who found a vulnerability (security researcher) and the company that developed the software (vendor). Both parties agree on a time that the vendor should provide the vulnerability patch. Only after the security update, the vulnerability information is made public. This information is usually shared in broad audience social media, such as Twitter and also by National Vulnerability Database) (NIST, 2019). This coordinate system holds the vendors accountable for the released software and also, acts as a practical way to inform users about vulnerable software.

Furthermore, the more widely used the software, the more the surface of attack is increased. This situation creates a constant need for securing and hardening Information Systems to avoid leaking private data and intellectual property, as well as to avoid damaging a company's reputation. According to Juniper Research (2017), the estimative of investment in cyber security is $135 billion by 2022, with a growth rate of 7.5% each year. There is also a growing concern with international espionage (Hjortdal, 2011, Lindsay, 2013). As seen in Deibert et al. (2009), some countries have clearly stated that cyberespionage is part of their strategic activities as a sovereign country. Preventing cyberattacks is a difficult task that can be tackled from various perspectives, for instance, from a software engineering standpoint, we can improve software in its development lifecycle to deliver better (and more secure) products; while from a cyber security standpoint, we can collect information to strategically act to avoid cyber attacks and vulnerability exploitation.

## 1.2 Publications

The list of publications that are part of the scope of this theses are:

- Queiroz, A., Keegan, B., and Mtenzi, F. (2017). Predicting software vulnerability using security discussion in social media. In *16th European Conference on Information Warfare and Security*, ECCWS, pages 628–634.

- Queiroz, A. L., Mckeever, S., and Keegan, B. (2019). Eavesdropping hackers: Detecting software vulnerability communication on social media using text mining. In *The 4th International Conference on Cyber-Technologies and Cyber-Systems*, pages 41–48.

- Queiroz, A. L., Mckeever, S., and Keegan, B. (2019). Detecting hacker threats: Performance of Word and Sentence Embedding models in identifying hacker communications. In *The 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science*, volume 2563, pages 116–127.

- Lima, A. Q. and Keegan, B. (2020). Chapter 3 - Challenges of using machine learning algorithms for cybersecurity: a study of threat-classification models applied to social media communication data. In Benson, V. and Mcalaney, J., editors, *Cyber Influence and Cognitive Threats*, pages 33–52.

- Queiroz, A. L., Keegan, B., and Mckeever, S. (2020). Moving Targets: Addressing Concept Drift in Supervised Models for Hacker Communication Detection. In *International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–7.

## 1.3 Contributions

The principal contribution of this thesis is the investigation of how machine-learning based models can be used to detect malicious communications related to software vulnerabilities in hacker forums and social media. The foremost goal is to identify the optimal configurations and techniques that make these models suitable for real-life application.

In the folloiwng, we present the contributions of this thesis in detail:

1. We provide a publicly available labelled dataset containing 10,000 messages related to vulnerabilities in software products and internet services. These messages were collected from different Internet domains (Surface, Deep and Dark Web), which includes 3 hacker forums, 1 Twitter and 1 Marketplace datasets. The labelling task was systematically performed by multiple labellers to guarantee that the labels are not biased by the subjectivity of the annotators.

2. We demonstrate, through empirical investigation, that the use of oversampling techniques in imbalanced datasets increases the recall of traditional classification models (aimed to detect software vulnerability communication) by 5% (on average).

3. We present the optimal techniques using Feature Reduction, Feature Extraction and Feature Selection to improve the efficiency of traditional classification models.

4. We demonstrate that specific-vocabulary language models are potentially better than general-vocabulary language models for detection of software vulnerability communication.

5. We demonstrate that Sentence Embedding language models acting as feature representation enhance the performance of classification models in downstream task.

6. We presented a study on deep learning-based classification models using text CNN and BiLSTM architectures. Both architectures have shown the best overall results for detection of software vulnerability communication, compared to traditional algorithms.

7. We demonstrate that models created using hacker forums content degrade in a period of less than 1-year, indicating Concept Drift.
8. We propose optimal methods for avoiding performance degradation of classification models over time, under Concept Drift effects.

## 1.4 Thesis Overview

The thesis is structured as in Figure 1.3 and described as follows:

- **Chapter 2** provides an overview of the principal concepts of text mining with focus on supervised learning and text classification. It describes the application of traditional classification and deep learning classification techniques for solving real-world problems that use unstructured textual data. Additionally, we discuss the importance of feature representations and classifier algorithms to create classification models using textual data. In conclusion, we provide a literature review of the supervised classification models used for the task of detecting malicious communications in hacker forums and social media.
- **Chapter 3** describes the sources of the 5 datasets used throughout this thesis. Additionally, we present the systematic approach taken for reducing the volume of messages, labelling and providing the final label. Moreover, we present the methodology used to perform the experiments in this thesis. This includes the selection of text classification algorithms and features representation, as well as the evaluation metrics and the feature reduction techniques.
- **Chapter 4** provides an investigation of models created with traditional classification algorithms and traditional features representations. The principal goal is to identify the optimal configuration of these models for detecting hacker communication regarding software vulnerabilities. The performance of these models is assessed and compared with multiple datasets. Additionally, we have investigated optimum approaches for dealing with imbalanced datasets and techniques

for optimising the classification model, for instance, document frequency reduction, feature selection and feature extraction. Furthermore, we compared two classification approaches, binary and multi-class classification for the detection task proposed in this thesis.

- **Chapter 5** evaluates Deep learning-based models in detecting software vulnerability communication. Firstly, we investigate the use of these Deep learning language model used as feature representation, for instance, Word and Sentence Embedding. Also, we compare the use of general-vocabulary and security-vocabulary Word Embedding for the proposed task. Furthermore, we investigate the performance of Sentence Embeddings model to build the classification model. Finally, we evaluate the use of two different Deep Learning architectures commonly used in Natural Language task, text CNN and an RNN-based architecture called BiLSTM.
- **Chapter 6** simulates the real-world application of these models by periodically evaluating its performance in 1-year period. With that, we investigate whether the performance of these models will decrease, indicating Concept Drift. Moreover, this evaluation is performed using three different datasets collected from Deep and Surface Web domains. Furthermore, we provide strategies to avoid performance decreasing, thus, prolonging the model's accuracy for a longer time.
- **Chapter 7** summarises the key contributions of this work, its limitations, and highlights opportunities for additional research

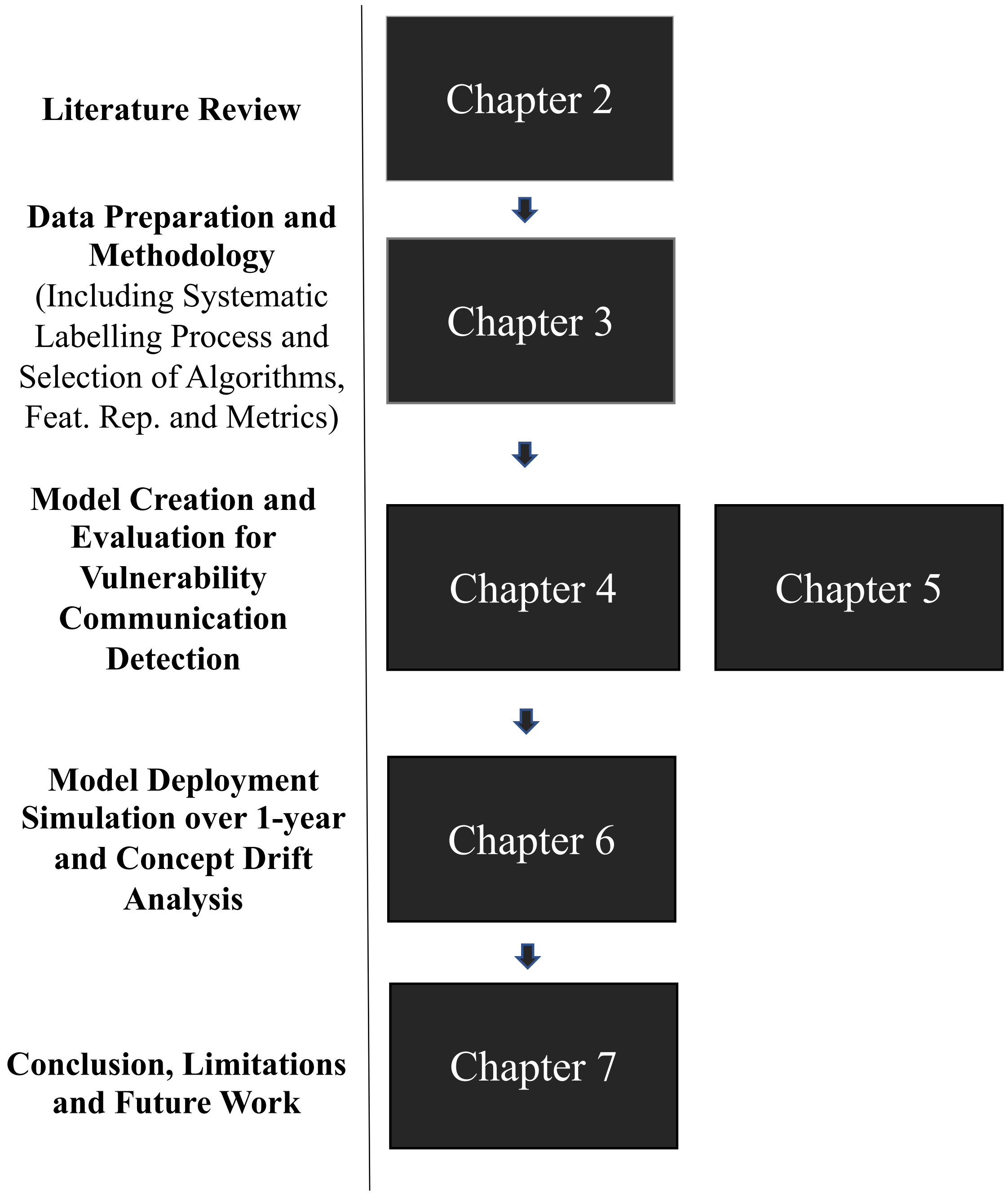

Literature Review
Chapter 2
Data Preparation and Methodology
(Including Systematic Labelling Process and Selection of Algorithms, Feat. Rep. and Metrics)
Chapter 3
Model Creation and Evaluation for Vulnerability Communication Detection
Chapter 4
Chapter 5
Model Deployment Simulation over 1-year and Concept Drift Analysis
Chapter 6
Conclusion, Limitations and Future Work
Chapter 7

Figure 1.3: Structure of the Thesis

CHAPTER 2

# Literature Review

This thesis aims to investigate techniques for automatic detection of software vulnerability communication in hacker forums and social media. Due to the vast amount of generated data on these channels, it is virtually infeasible for Cyber Threat Intelligence (CTI) teams to analyse them through manual inspection. Thus, appropriate methods for allowing automatic language understanding should be considered to provide a distinction between threat and non-threat messages in hacker communication.

Recently, data-driven approaches have been used to solve a variety of tasks using large quantities of data. Encouraged by this, we aim to review a range of language models and supervised learning-based algorithms applied to text classification tasks. These methods have shown promising results in traditional classification tasks, for instance, news categorisation (Mittermayer and Knolmayer, 2006), sentiment analysis (Jianqiang and Xiaolin, 2017, Ren et al., 2016, Tang et al., 2015), and spam filtering (Feng et al., 2016, Lee and Kang, 2019), among others.

Therefore, this chapter begins by introducing the process of creating a supervised model in Section 2.1, which includes the challenges of acquiring and labelling data in the security research domain (Section 2.1.1) and a review of evaluation designs and metrics used to validate the model (Section 2.1.2). Furthermore, in Section 2.2, we present the state-of-the-art machine learning techniques and algorithms used for automatic categorisation of textual data, including traditional- and deep learning-based algorithms, and unsupervised feature representations (language models).

Finally, in Section 2.3, we provide a review on cyber security research that uses text classification techniques. In this overview, we compare works regarding three overlapping aspects: Labelling task, Domain-specific features and algorithms. Additionally, in Section 2.3.2, we discuss the application of unsupervised algorithms (language models) for clustering similar words and jargon in hacker forums.

## 2.1 Fundamentals of Data-driven Approach

In this thesis, we focus on supervised data-driven approaches applied to classification of textual content, commonly called "text classification". This is an interdisciplinary approach which borrows from other domains, such as computational linguistics, statistics, machine learning and data mining. The main challenge of this method is to provide comprehension of unstructured information, for instance, text and sentences, which is easily perceived by humans, but difficult for machines. Additionally, these methods have been frequently used for solving several traditional Natural Language Processing (NLP) problems, for instance, Information Retrieval (IR), Information Extraction (IE), Named-entity Recognition (NER).

Therefore, the process of creating supervised classification models requires the *training* of an algorithm. This training uses a labelled (categorised) dataset, where each instance belongs to a class. There are three principal steps to be followed to create these models:

1. **Pre-processing** - the sentence is split into small parts, such as words or characters (*tokenisation*); Afterwards, the words that do not carry much information are removed, for instance, pronouns and articles (*filtering*). Also, these tokens are represented by weights, which the values are adjusted to a defined range of

numbers (*Normalisation*).

2. **Learning** - the algorithm and the some individual/informative characteristics of data (commonly known as features) are combined to train the model.

3. **Evaluation** (or test phase) - the model is evaluated using unseen data (data that the model has not used in the training phase). In the end, the model has "learnt" all given examples, thus, it is can be deployed in a real production environment to predict further unseen instances. The entire process can be seen in figure 2.1.

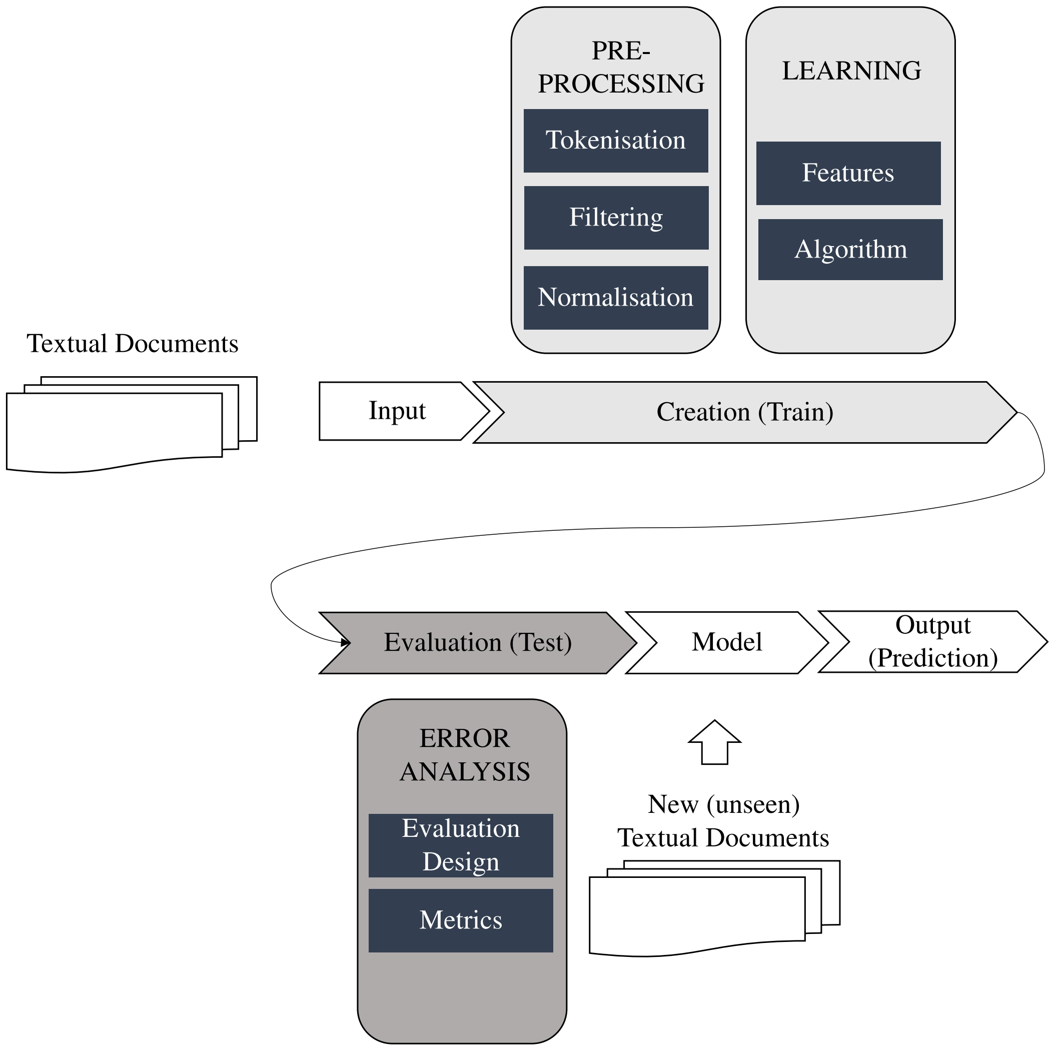


Figure 2.1: Basic Training Process of Machine Learning Models for Text Classification

### 2.1.1 Data Acquisition and Labelling

Labelled data is a requirement for supervised approaches. To perform the *labelling* of data (annotation), there are two commonly used approaches:

- **Expert labelling** - This approach can be expensive due to the need for the right skill set and human training to carry out the task. However, it is the most suitable method when the task requires domain knowledge to guarantee quality labels (Welinder and Perona, 2010).

- **Crowdsourcing labelling** - In contrast to the previous approach, Crowdsourcing is frequently used for larger amounts of data. It can be less expensive and less time consuming, as the workload is generally distributed among multiples of labellers provided by the crowdsourcing platform. However, the drawback is a potential lack of reliability as the labeller might not have the skill needed to perform the annotation task (Wang and Zhou, 2015).

Regarding the related literature in data-driven approaches for cyber security, we have mainly seen authors using the Expert Labelling approaches, in which the authors are responsible for assigning the labels to the dataset. Moreover, we have observed multiple approaches for performing this task, for instance, some authors have used keywords to help with the labelling annotation task (Deliu et al., 2017, Deliu et al., 2018), while others have used external sources to properly assign the labels (Mulwad et al., 2011, Sabottke et al., 2015). Additionally, it is worth to mention that, there is a lack of a reliable labelled dataset (commonly called gold standard or ground truth dataset) in this research domain, which prevents the comparison between model and the reproducibility of works, as related by Benjamin et al. (2015), Nunes et al. (2016), Sabottke et al. (2015). A more detailed discussion on this issue is presented in Section 2.3.1.

### 2.1.2 Evaluation Design and Performance Metrics

The evaluation of supervised classification models is based on unseen/untrained instances, commonly referred to as the *test samples*. The results of the classification model are presented in a contingency table called *confusion matrix*. In Table 2.1, we see an example of a confusion matrix table for binary classification models, where the classes are positive and negative. The values of the cells are given as follows:

Table 2.1: Confusion Matrix For Binary Classification

| | | **Predicted Label** | | |
|---|---|---|---|---|
| | | **Negative** | **Positive** | **Total** |
| **True Label** | **Negative** | TN | FP | TN+FP |
| | **Positive** | FN | TP | FN+TP |
| | **Total** | TN+FN | FP+TP | |

- True Positive (TP) represents the positive instances that are correctly predicted as positive
- False Positive (FP) refers to the negative instances that are falsely predicted as positive
- True Negative (TN) is the negative instances are correctly predicted as negative
- False Negative (FN) stands for the positive instances are falsely predicted as negative

Furthermore, a commonly used approach for assessing the performance of models is the K-fold cross-validation evaluation design (with $K = 10$ folds) (Chen et al., 2017a, Deliu et al., 2017, Lippmann et al., 2016, Nunes et al., 2016). Through this approach, the dataset is randomly partitioned into K equally sized folds preserving the proportion of positive and negative instance for each fold, according to the original dataset (*stratification*). Then, the model is trained K times, with each fold held back exactly once to be used for evaluation (test) as shown in Figure 2.2. This evaluation provides a good estimation of how well the model will perform when deployed (Kelleher et al., 2015).

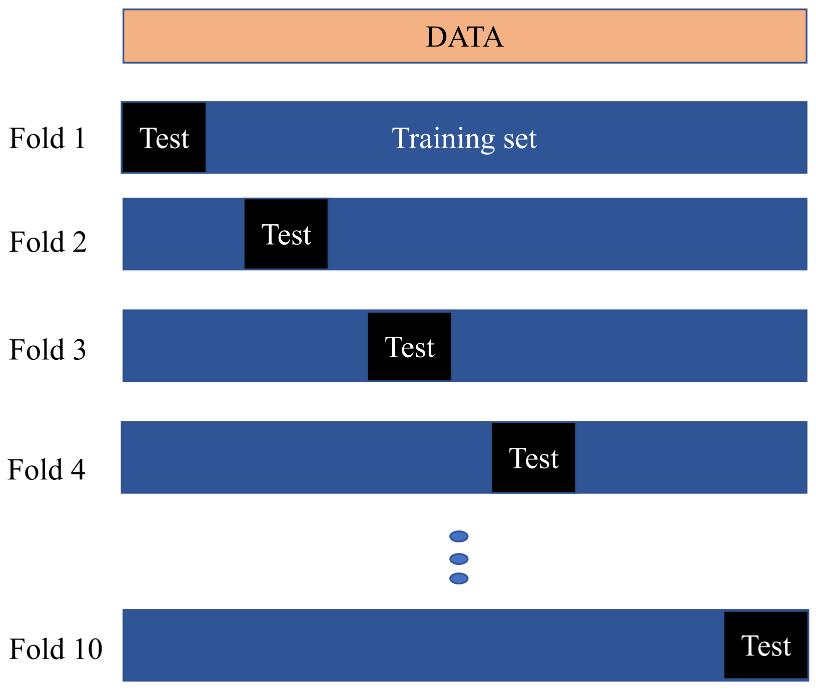


Figure 2.2: K-fold Cross-validation Design

## 2.2 Text Classification Models

Text classification is a branch of classification domain that aims to predict or categorise textual documents. From a mathematical viewpoint, we are modelling an approximation function $f(x)$ from input samples $x$ to its output $y$. At the core of this modelling, we use a (1) learning algorithm, and (2) text/features representation. In this section, we provide a historic review of the so-called features representation, which is the method for translating the textual messages to be further used as input for learning-based algorithms. Furthermore, we provide a review of the learning algorithms used in text classification, starting with traditional linear algorithms, and moving onto more robust Neural Network-based architectures.

### 2.2.1 Overview of Features Representations

In text classification approaches, parts of the message can be represented either by words, characters or symbols (tokens). Then, these tokens are transformed into a representation which can be understood or interpretated by an algorithm. These representations can be used as *Features* for the model, and are of paramount importance for classification and computational performance of the models (Allahyari et al., 2017). In this thesis, we use the term *Features* interchangeably with *Text/Features Representation*, or simply *Word Features*.

Features representation can be used for retaining contextual information of messages. The earliest form of representing these messages in language system was through the so-called Vector Space Model (VSM), for which *bag-of-words, word n-grams and char n-grams* are the popular VSMs. Recently, there has been a surge in other types of features representation, commonly called Language Models, more robust than the traditional VSM models, it can retain the semantic and syntactic relationship of words and sentences. In the following, we present an overview of these approaches by highlighting their properties, starting with common examples of the traditional Vector Space Model (VSM), moving onto more robust models which are divided into *Word Level Distributed Representation* and *Sentence Level Distributed Representation*.

#### Vector Space Model (VSM)

The most common and earliest form of representing word features in text classification is through a Vector Space Model (VSM). VSMs were introduced in information retrieval by (Salton et al., 1975), and are still widely used in text mining algorithms and retrieval systems. In this scheme, the feature on a VSM model is represented by weights, which term frequency (TF), and the Inverse Frequency (TF-IDF) are common representations (Salton and Buckley, 1988).

A traditional example of VSM is the *bag-of-words* (BoW) representation, used in a large number of text classification tasks (Chen et al., 2017a, Lee and Kang, 2019, Liu, 2017, Nunes et al., 2016, Wu et al., 2017). In this representation, the document/sentence is split in words that represent a single feature of the classifier. Another common representation is the n-gram, also used in text classification model, where the *n* of n-gram is used to define the number of sequential tokens representing a single feature, where the tokens can be entire words (*words n-gram*) or characters (*char n-gram*).

The principal problem in VSM representations is that they might end up as large vectors (or high-dimensional) and a large number of zeros (sparse), depending on the size of the vocabulary. This problem is also known as "Curse of Dimensionality" which might damage the performance of models due to an imbalanced of the number of features (high-dimensionality) and the size of the dataset (number of examples) (Bellman et al., 1957). To soften the high-dimensionality problem, non-informative words are often removed, where the goal is to reduce the noise by removing words such as pronouns and articles, while also reducing the dimensionality of the vector. Other tasks, such as Lemmatisation and Stemming, is used to reduce the word to its core (e.g., the words "hacking", "hacked" and "hacker" are reduced to same token "hack"), are also used to reduce the dimensionality of VSMs. There are more robust techniques for reducing the dimensionality of VSMs, for instance, features selection and feature extraction, which we discuss on Chapter 4, Section 4.7.

These models are simple to implement and also provide good results in many NLP tasks. However, they are limited with regards to the information retained in the vector values. Recently, research in language representation has provided forms of enhancing the capability of these traditional features representations by allowing them to provide information on the semantic and syntactic meaning of words (or sentences). These models allow assessing the relationship between the words by simple mathematical operations. Such characteristics have been seen as a breakthrough into the language representation domain and have also enhanced traditional VSMs models. In the following, we provide an overview of models that provide such characteristics, starting by the Word Level Distributed representation, or Words Embeddings (WEMB), and afterwards, the Sentence Level Distributed Representation, or Sentence Embedding (SEMB).

### Word Level Distributed Representation

Word2vec language model introduced by (Mikolov et al., 2013) is a language model that allows assessing the semantic and syntactic relation of words through simple algebraic operation, for instance, the operation vector("King") - vector("Man") + vector("Woman") would result in vector values close to the vector representation of ("Queen"). The same operation can be performed throughout all words of a corpus where the model was trained.

This model has been successfully used in several NLP tasks such as Machine Translation, Information Retrieval and Question Answering systems, and also has shown promise in improving accuracy on several downstream classification accuracy tasks (Kameswara Sarma et al., 2018, Roy et al., 2017, Yang et al., 2018). *Word2vec* is also presented in two different Neural Network-based architectures, the CBOW and Skip-gram. They differ on the performed task, where the former performs a prediction of the current word given a surrounding context, whereas the latter predicts the surrounding words given the current word. During this task, a dense vector representation is calculated for each input word (embedding), where the resulting vector is used as input in algorithms for a downstream classification task.

There is also another method called *Glove*, introduced by Pennington et al. (2014). Similarly to *Word2vec*, this model is often used in downstream classification and word clustering tasks. However, different from the previous, *Glove* is not based on Neural Network architecture, but on a matrix factorisation of co-occurrence of words, which makes its training more computationally expensive. Both models have a similar advantage regarding the mathematical operation on similar words, but also the same drawback, they lack the ability to adequately capture word polysemy (different meanings for unique word). Moreover, *Glove* has been providing better results than

*Word2vec* on some NLP task (Pennington et al., 2014).

### Sentence Level Distributed Representation

Sentence Embedding (SEMB) is focused on providing a fixed-length dense vector that represents an entire variable-length sentence while keeping similar sentences alongside in a vector space. This model, as the name suggests, is focused on providing a vector encoding for the entire sentence instead of encoding each unique words of the sentence. The main characteristics of SEMBs are the ability to differentiate between sentences with the same words, placed in a different order. For instance, the sentence "Play hard, don't study!" and "Study hard, don't play!" have different meanings, and this difference is encoded by SEMB model, but is lacking on previously seen models (VSMs and WEMB), which would treat these phrases as being equal.

There are several proposed architectures for SEMB models, each one is trained in a different task and with different types of data. For instance, the authors Le and Mikolov (2014) proposed a SEMB model based on unsupervised learning approach called *Sent2vec*. This model follows the Word2vec approach, although it adds a new vector that maps the resulting Word Embedding to a single vector that represents the entire sentence. Another model introduced by Kiros et al. (2015) called Skip-Thought is based on Wor2vec Skip-Gram training task, which uses a central sentence to reconstruct the surrounding sentence. Additionally, this model uses an encoder-decoder architecture to provide better use of the computational resource.

Beyond the unsupervised approaches presented so far, there are two other forms for creating SEMB. One uses supervised learning and the other uses both supervised and unsupervised approaches (multi-task). An example of a supervised SEMB was introduced by Conneau et al. (2017). This model achieved better performance in transfer learning tasks compared to other two state-of-the-art unsupervised SEMB models, *SkipThought* and *FastSent*. The so-called *Infersent* model is trained in a supervised task called Natural Language Inference (NLI), which, according to Conneau et al. (2017), provides better embedding representation due to the high-levels of understanding and reasoning involved in this task. Furthermore, using a multi-task approach, the so-called Universal Sentence Encoder model (*SentEncoder*) (Cer et al., 2018), utilises a variety of sources in its unsupervised module, including Wikipedia, question-answer pages and web news, and in its supervised module used the labelled Stanford Natural Language Inference (SNLI) corpus, similarly to *Inferset*. Language model research has advanced in a fast-paced manner. It continues to evolve during the writing of this thesis. As a consequence, we are not covering all the newest language models architectures.

#### Transformer-based Models

Currently, the new generation of language models has been built with the so-called Transformer architecture (Vaswani et al., 2017). This architecture can also process the input information using parallel computer processing rather than sequentially, resulting in fast training time compared to non-transformer models. This characteristic is due to the use of the *Attention* mechanism. Moreover, Transformer models perform better with long-range dependencies compared to RNN-based and CNN-based sequence-to-sequence language models.

The characteristics described combined with larger datasets have resulted in state-of-the-art language models for NLP tasks. For example, popular Transformer language models include BERT (Devlin et al., 2018) by Google, GPT2 (Radford et al., 2018) and GPT3 (Brown et al., 2020) by OpenAI, with a total number of parameters of 340M, 1.5B and 175B, respectively. However, high volumes of data require a large computational capacity to train such models. For instance, GPT3 model needs an infrastructure of 285,000 CPUs and 10,000 GPUs to be trained, which costs more than $ 4.6M to achieve the computer power requirements on cloud platforms (INFO-Q, 2020).

### 2.2.2 Overview of Traditional Algorithms

The algorithm is the basis of classification models. There are a variety of algorithms that can be used to solve classification tasks. Among them, we commonly see K-nearest Neighbour (Cover and Hart, 1967), Logistic Regression (Zhang and Oles, 2001), Decision Trees, e.g., ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), and Naïve Bayes (McCallum and Nigam, 1998).

Although technically, all these algorithms can be used in text classification tasks, SVM and Naïve Bayes are the most commonly used. Studies indicate that the former algorithm is among the best text classifiers and also, the most suitable for high-dimensional features set (Dumais et al., 1998, Joachims, 1998, Yang and Liu, 1999), whereas the latter is a simple but effective Bayesian learning method that has been providing good performance in text classification problems (Domingos and Pazzani, 1997). These algorithms have been applied to a variate of text classification tasks, for instance, sentiment analysis, spam detection, news categorisation (Feng et al., 2016, Jianqiang and Xiaolin, 2017, Liu, 2017, Mittermayer and Knolmayer, 2006).

One of the principal characteristics of traditional algorithms, including SVM and Naïve Bayes, is the need for predefined features. In other words, the selection of features for these algorithms requires human with knowledge in the specific domain of application. Although it requires an extra effort to find which features would improve the performance, it allows us to use them to explain the resulting classification of the model. This characteristic is not present in Neural Network-based Algorithms to date.

### 2.2.3 Overview of Neural Network-based Algorithms

Recently, we have been experiencing the popularisation of Artificial Neural Network (ANN). These algorithms have an inspiration in the biological process of human brain, in which artificial neurons are used for learning patterns in past historic data to predict further unseen information. This approach has been providing some of the state-of-the-art results in classification tasks, although, in contrast to traditional algorithms, it is not possible to explain the classification results using the classifier's features. This is due to the non-requirement of hand-crafted features, thus it acts as a "Blackbox" system.

Another characteristic of ANNs is that they require a large quantity of data to avoid overfitting. As a consequence, some traditional algorithms such as SVM can be more effective than ANNs with small data sets, as they use a less computational resource and achieve better accuracy (Liu et al., 2017). However, to partially overcome the overfitting problem of ANNs, Hinton et al. (2006) has proposed a training method called layer-wise-greedy-learning, which has provided the beginning of Deep Learning (DL) approaches. The basic idea of DL algorithms is to expand the neuron layers of ANNs, which they called hidden layers. This procedure allows obtaining a compact representation of the data, reducing the possibility of overfitting and achieving faster convergence (training), although it still more computational resource intensive than based on traditional methods (Liu et al., 2017).

DL algorithms have been providing state-of-the-art results in several domains, for instance, image classification, language translation, documents retrieval, abusive comments detection (Chen et al., 2019, Huang et al., 2014, Kim, 2014, Palaz et al., 2015). There are two promising examples of DL algorithms used in text classification, so-called text CNN and RNN. CNN architecture started its foundations in LeCun et al. (1990) papers and was improved in Lecun et al. (1998), and the focus was in image applications. Since then, a large number of network architectures have been proposed, among then are Alexnet (Russakovsky et al., 2014), VGGNet (Simonyan and Zisserman, 2014), GoogleNet (Szegedy et al., 2014), ResNet (He et al., 2015). Although CNNs are mostly applied in image recognition tasks, this architecture has been adapted for text classification and has provided good results in Natural Language tasks, including sentiment analysis and question classification Kim (2014).

Another commonly used algorithm for Natural Language tasks is the Recurrent Neural Network (RNN). In contrast to CNN, this algorithm is characterised by the capability of mapping sequential data and its temporal dependencies. Audio, video and text are examples of sequential data (data that depend on previous information). The temporal information of data is then mapped through cyclical connections. This allows the updating current state of an RNN cell with past dependencies Yu et al. (2019). However, RNN is not suitable for mapping long-term dependencies within data, which incurs in a well-known problem called *vanish gradient* (Hochreiter, 1998). To overcome this limitation, an RNN-based architecture was proposed by Hochreiter and Schmidhuber (1997), called Long Short-Term Memory (LSTM). As opposed to the basic recurrent unit, the LSTM architecture introduces a "gate" mechanism, which allows the architecture to decide which data is stored and removed from its current cell state. LSTM is used in the majority of the state-of-the-art results and has performed better compared to the basic RNN (Tang et al., 2015, Wang et al., 2015).

## 2.3 Text Classification in Cyber Security

In this section, we provide the state of the art of studies applied to the detection of malicious communication shared in social media channels. We highlight that the majority of the work discussed in this chapter is not completely related to the principal goal of this thesis. However, in some extent, it may overlap with our purpose by including a more generic objective for example, the detection of **malicious conversation**, **hacker attacks discussion**, **malicious cyber communication** or **malicious software attachments**. Therefore, all research presented in this section uses data-driven approaches for detection of malicious communication in social media.

To introduce these studies, we divided this section into two parts. The first provides a discussion of *Supervised Classification* approaches using machine learning classification algorithms. The second provides a discussion regarding some *Unsupervised* approaches to support cyber security investigation on social media channels.

### 2.3.1 Supervised Approaches

We have focused on three main aspects of data-driven methods: The labelling task; Domain-Specific Features; and algorithms choice, so that we might compare and discuss the different approaches thoroughly.

#### A) Dataset and Labelling Task

As discussed in Section 2.1.1, supervised learning approaches require a labelled dataset. The related literature categorises these labels into two classes: the **positive or target** class, for messages representing malicious communication content, and **negative or non-target** class, for messages that do not represent a threat or hacker malicious content.

However, the process of labelling this type of data, especially those related to cyber security, is difficult due to the need of experts that can understand the jargon and technical information of these messages (Benjamin et al., 2015, Sabottke et al., 2015, Trabelsi et al., 2015). It is also expensive and time-consuming, not to mention the difficulty of collecting these messages in hacker forums, as these channels have an anti-crawling mechanism to prevent the messages being collected by automatic tools (crawlers) (Fu et al., 2010).

As a result, we have identified that, in these works, the authors are using their expertise to annotate the labels. Additionally, it has been noticed that the use of crowdsourcing service providers to perform the labelling is not common in the security domain. Furthermore, we have identified three different methods that the authors have used to provide labels to the dataset messages, which we discuss below:

**Human Expert labelling:** In this case, the authors used expert knowledge to annotate the labels of the messages. As seen in Nunes et al. (2016), they manually labelled only 25% of the dataset, while the unlabelled part, 75%, was annotated by a technique called co-training (CT). Although this technique has been used to enhance the performance of classification models by increasing the quantity of labelled data, the authors fail to adequately explore how reliable the assigned labels are for this specific problem.

**Labelling by keyword matching:** In this case, the authors assumed that the messages with specific keywords could be categorised either as positive or negative classes. As seen in Deliu et al. (2017) and Deliu et al. (2018), they pre-defined a security-keywords list, where messages containing at least one word in common with the list are assigned a positive class label. On the other hand, the negative class messages are those which do not match any word from the mentioned list. As an example, the security list provides words, such as *adware, antivirus, backdoor, botnet, exploit, cve, exploit, firewall, hijack, infect, keylogger, security, shell code, spam, crypter, ddos, password*. Whereas, the negative messages include words relating to sport, movies, music, etc.

Also, in Portnoff et al. (2017), where the aim of the work is to detect whether the posted message is related to the trade of hacking products or not, they annotated the messages according to the description of the thread topic of the forum. If the thread topic uses words that refers to the trade of products, for instance, *buy*, *sell*, *currency*, all messages under this topic were assigned as being from a positive class, whereas for topics with words not related to trade, the messages were assigned as being from a negative class. Similarly, in Lippmann et al. (2016), instead of using keywords, they have assigned the same category to messages under a specific social media topic. For instance, on Reddit social media dataset, all messages under reverse engineering, security, malware, and blackhat were assigned as positive, whereas messages under topics not related to computer security, such as astronomy, electronics, beer, biology, music, and movies were assigned as negative class.

We believe these works are naively assuming that keywords can guarantee the real category of messages. If it were true, we would not need robust algorithms to distinguish the meaning of these messages. As outlined later in this thesis (Chapter 3), the same keywords might appear in both categories of messages, in which case, the distinction is made by understating the entire contextual information.

**Labelling by external reference:** In this case, the authors have used an external reference to assign the labels on dataset instances. As seen in Sabottke et al. (2015), the authors have created a model where the goal is to predict whether a vulnerability is being exploited or not. For labelling the dataset, they have used Symantec's anti-virus and intrusion-protection signatures report, which mentions what vulnerabilities have been exploited. The authors have claimed that they use this source because there is a lack of ground truth dataset for study classification models in this area. Additionally, they complain that these sources do not cover all vulnerabilities, as well as leaving Linux products out of the scope of the report.

Similarly, in Mulwad et al. (2011), using an external source as part of their framework for collecting security information posted on social media platforms; the authors trained a model using vulnerability descriptions of National Vulnerability Database (NVD) as positive instances, and technical messages found on CNET web portal, as negative instances. With this model, the authors suggest that it would be useful for monitoring potential hackers attack and vulnerability mentions in social media channels and chat rooms. However, the authors fail to adequately provide the resulting performance of the model in the mentioned social media sources.

### B) Domain Specific Features

Some works have been using social media domain-specific features to increase the performance of the classifier. In Cherqi et al. (2018), the authors created a model for detecting *hacker related* and *non-hacker related* messages in marketplaces on Dark Web. They have used domain-specific features, such as the origin of products, the destination of products and rating of products. This information is domain-specific and cannot be seen in all hacker forums, or on other social media channels. Moreover, in Benjamin and Chen (2012), the authors have also used specific metadata called *hacker-rating* (rate the contribution of the user) to enhance the performance of the classification models applied to hacker forum communication.

Another example of the use of domain-specific features can be seen in Sabottke et al. (2015), the author has used the number of re-tweets on Twitter social media to improve classification performance in predicting the next software exploited vulnerability. However, similarly to Cherqi et al. (2018), the feature can only be used in a Twitter dataset, and as the previous work, it is not part of other social media, especially hacker forums.

In summary, some types of information are domain-specific and are not found on every social media platform. As a result, we have chosen not to add domain-specific features in our experiments, as we are using 5 different datasets collected from different sources. We acknowledge that domain-specific features might be used to enhance the performance of classifiers, however, it would not be fair for comparison between models created in different sources. For this reason, we are only using the words (and characters) of sentences as features, as they are the most informative feature and are present in all datasets.

**C) Classification algorithms**

The selection of algorithms in cyber security-related literature does not differ from other text mining approaches, where the traditional SVM is one of the top choices for text mining tasks. SVM has been showing promising results in detecting transaction in Dark Web forums (Portnoff et al., 2017), cyber security-related events in social media (Deliu et al., 2018), and malware, exploitation and vulnerability communication (Nunes et al., 2016, Sabottke et al., 2015). Therefore, deep learning-based approaches have also shown good results using RNN and CNN architectures, as presented by Grisham et al. (2017) in a mobile malware attachment detection and by Deliu et al. (2017) in a security-related communication detection. In the latter work, the author has provided a comparison of CNN and traditional SVM algorithms, where both algorithms have performed similarly well in the same task.

As we have previously discussed at the beginning of section 2.3, these techniques are applied to a broad context of hacker threat, which in some cases overlaps with our purpose (software vulnerability communication). Although these approaches indicate promising results, we need to perform a thorough analysis of these algorithms and features representation focused on our specific purpose and assure that we provide a sound dataset with quality labels.

### 2.3.2 Unsupervised Approaches

In contrast to supervised approaches, unsupervised does not require a labelled dataset. However, unsupervised approaches are not commonly used for classification as its evaluation is done mostly by qualitative analysis. As a consequence, unsupervised methods are frequently used to cluster similar information. An Example of a model created using unsupervised methods is the so-called language models, which is commonly used in hacker forum investigation as seen in Benjamin and Chen (2015). In this work, the authors have used a language model to capture the meaning of hacking-specific language, jargons and concepts. To perform this task, they have built two models, using CBOW and Skip-Gram architecture (seen in Section 2.2.1). The resulting model was able to capture the similarity of the 10 popular hacker terms, i.e., botnet, RAT, card, logger, crypter, rootkit, salt, binder, dork, vulnerability. Moreover, CBOW with Negative Sampling (NS) has shown the best result using an evaluation metric called precision-at-10, which the results is given based on the similarity of the first 10 output words with the input word. The similarity of the output and input is based on the experience and subjectivity of the research authors.

A similar work presented by Zhao et al. (2016), applied the two language models, Word2vec and Latent Dirichlet Allocation (LDA) to a Chinese hacker forum for identifying similar jargons with related cyber security terms. The results show that Word2vec models can overcome LDA in the task of similarity. This work has also used the same metric as Benjamin and Chen (2015) (precision-at-10). Additionally, the authors have highlighted the difficulties of analysing hacker forums, and have also argued that the major difficulty for cyber security researchers and investigators is usually the unfamiliarity with novel terms due to the fast-pace emerging of new hacking concepts.

Furthermore, the work done by Roy et al. (2017) highlight that specialised Word Embedding, that is, a Word Embedding build on top specific-language vocabulary (i.e., using a corpus with cyber security vocabulary), might not produce high-quality embedding due to the lack of co-occurrence words within the dataset instances. With that in mind, they have provided an annotation framework called Word Annotation Embedding (WAE) to deal with this limitation. Their approach has improved over the common approach. However, it relies on the ability (and availability) of the human annotator. Additionally, there is also the added task of labelling before the creation of the embedding, making the creation of unsupervised models more labour intensive.

In summary, large part of the work in this domain used unsupervised approaches to understanding hacker language by clustering new related concepts and jargons. However, there are other forms of using these language models, such as using as features representation in *downstream classification tasks*. In other words, we can use pre-trained language models in combination with supervised algorithms as they provide classification improvement in a variety of language problem (Amir et al., 2016, Perone et al., 2018, Wang et al., 2015, Wieting et al., 2015). Therefore, in this thesis, we focus on the use of these unsupervised models in combination with supervised classification algorithms, to investigate whether this combination enhances the performance of models applied to the detection of software vulnerability communication.

## 2.4 Rule-based Systems

A classic and commonly used rule-based system is the so-called *Expert System* (ES) (Russell and Norvig, 2009, p. 633). In this system, a set of rules is created by a human expert to build the Knowledge Base (KB). Based on the interaction of the KB rules and the so-called *Inference Engine*, the system can perform actions, such as reasoning, deductions, choices and predictions.

Comparing ES with machine learning-based approaches, the main difference lies in how the set of rules is specified. In ES, the rules are tailored by domain-experts, whereas in machine learning approaches, the rules are discovered by learning-based algorithms through patterns in data, without necessarily requiring human intervention to define them. Although supervised learning approaches might also require a human expert to categorise the dataset (labelling phase), this step can be replaced by crowdsourced services, such as Amazon Mechanical Turk or similar labelling platforms (Wang and Zhou, 2015, Welinder and Perona, 2010).

Both approaches have been used in NLP tasks, although, machine learning has been the most commonly used approach nowadays. Additionally, studies have shown a slight advantage of using machine learning approaches instead of the ES. As seen in Tan et al. (2018), the authors compared two medical NLP system aimed to detect the occurrence of Low Back Pain (LBP) diagnosis on radiology reports, one built as an ES and the other as a machine learning system. The authors presented that the machine learning system provides the best average recall, 94%, and Area Under the Curve (AUC), 98%, compared to 84% and 90% achieved by ES. In another publication, the authors Tiftikci et al. (2019) have also compared both methods. They provided a combination of three machine learning algorithms (CNN, Bi-LSTM and CRF) to identify Adverse Drug Reaction (ADR) in drug labels. This approache achieved 77% of F1-score compared to 67.4% achieved by the ES in the same task.

As a result, for this research, we have decided to pursue the machine learning approach as they (1) provide better results when compared to ruled-based systems, and more importantly, (2) there is no need to handcraft rules as required in ES approaches. Moreover, machine learning algorithms are built to discover the rules by themselves, which reduce human intervention and effort to build a classification system.

## 2.5 Summary and Conclusion

In this chapter, we outlined the basic process of creating supervised classification models for text classification problems, which includes three basic steps: Pre-processing, Learning (training) and Evaluation (testing). Afterwards, in Section 2.1.1, we highlighted the importance of a labelled datasets for supervised approaches and outlined the two commonly used strategies for collecting these labels: Expert labelling and Crowdsource labelling. In sequence, in Section 2.1.2, we outline the main metrics and evaluation designs taken to evaluate the performance of these models.

Furthermore, we reviewed one of the principal components of text classification, Feature Representation models. We started by describing the early VSMs (*bag-of-words and n-grams*) as well as more robust language models (Word and Sentence Level Distributed representation). In the latter, we compared the main advantages and limitation with regards to the retention of contextual information of words and sentences. Afterwards, we provided an overview of the commonly used algorithms for text classification, including traditional and Deep Learning-based algorithms. We have noticed that among the traditional algorithms, SVM is one of the top choices for text classification due to its ability to handle a large number of features. However, several Neural network architectures, for instance, CNNs and RNNs, have been adapted for solving NLP problems, achieving promising results.

Finally, we reviewed the approaches taken for classification models in cyber security. To overview these works, we compare them based on the common aspects of the creation of supervised models, for instance, the labelling task, features and algorithms. As a result, we have identified two gaps in these approaches within security research domains: (1) there is a lack of Ground Truth dataset for comparison and research reproducibility purposes, and (2) there is a lack of systematic approach for dealing with ambiguous messages and the subjectivity of labellers in the labelling task. To address this problem, we propose a systematic labelling approach in Chapter 3.

CHAPTER 3

# Dataset and Methodology

The purpose of this research is to investigate techniques for automatic identification of software-security communication posted on hacker forums and social media channels. To achieve this goal, we have applied supervised learning approaches to distinguish the *target* (related to software security) from *non-target* messages (non-related to software security). Thus, the target communication can be used in further countermeasures against hacker threats. The terms target and non-target are used throughout this thesis to refer to these different classes of messages.

Moreover, given the rapid development data-driven techniques for text analytic discussed in Chapter 2, we propose to cover a range of these approaches for creating classification models, which includes the use of traditional- and deep learning-based classification algorithms combined with different types of supervised and unsupervised feature representations.

The methodology and datasets described here are the basis for the experiments in Chapters 4, 5 and 6 which, in summary, consists of:

- Identifying the best configuration of traditional algorithms and feature representations for automatic identification of software-security communication posted on hacker forums and social media channels.

- Analyse strategies that provide efficiency and better use of these models, such as feature reduction, feature selection and feature extraction.
- Identifying the best configuration of Deep learning-based algorithms and robust feature representations for the automatic identification of software-security communication posted on hacker forums and social media channels.
- Evaluate the application of these models in a variety of sources, for instance, forums and marketplace on Deep Web, and microblogs on Surface Web.
- Evaluate the robustness of these models in a real-life application under concept drift effects.

In this chapter, in Section 3.1 we describe the original datasets and types of sources used for performing this experiment (forums and social media). In addition, Section 3.2 describes the details and principal characteristics of each source for the hacker community. As these datasets do not contain labels, in Section 3.3, we describe the steps taken for preparing them for expert annotation (labelling), which is also a necessary step for supervised learning approaches.

Lastly, in Section 3.4, we describe the methodology to create the models considering the combination of classification algorithms and feature representations. Also, we introduce the metrics for evaluating these models in our specific task, followed by Section 4.5 Summary and Conclusion.

## 3.1 Original Datasets Collection

The principal purpose of this research is to investigate forms of automatically detecting software-vulnerability communication on social media through supervised classification models. As a result, we have opted to follow a data-driven approach, as this method has been providing outstanding results in NLP tasks (as seen in Chapter 2).

Moreover, with data-driven approaches, it is possible to use examples of real-world messages to produce suitable classification models that can be used to distinguish the target from non-target messages. To have a representative number of messages (examples) that reflect our research purpose (software-vulnerability communication), we have selected 5 datasets from representative hacker communication channels, which cover commonly used social media and hacker forums from three domains: Deep Web, Dark Web and Surface Web.

The criterion for selecting these datasets was determined by the existence of a common channel where users interact to disclose and acquire knowledge regarding software vulnerability. Also, by using this range of datasets, we want to ensure that our classification model provides generalisable results to a broad range of sources. These sources are referred to in the following chapters of this work as D1, D2, D3, D4 and D5 and represent Hacker forum, marketplaces and Twitter security expert posts, from Deep Web, Dark Web and Surface Web, respectively. A textual description regarding the purpose and relevance of each dataset can be seen in Section 3.2, whilst the number of messages in each original set, type and source are summarised in Table 3.1.

Table 3.1: Original Dataset Collection Summary

| *ID* | *Source* | *Type* | *No. of msgs.* | *Published in* |
|---|---|---|---|---|
| **D1** | Hacker Forum | Deep web | 44,752 | Samtani et al. (2016) |
| **D2** | Twitter | Surface web | 11,832 | Queiroz et al. (2017) |
| **D3** | Marketplace | Dark web | 91,463 | Samtani et al. (2016) |
| **D4** | Hacker Forum | Deep web | 8,699 | Samtani et al. (2016) |
| **D5** | Hacker Forum | Deep web | 36,684 | Samtani et al. (2016) |
| **TOTAL** | | | 193,430 | |

The datasets D1, D3, D4 and D5 contain, respectively, 44,752, 91,463, 8,699, 36,684 posted messages and were originally published in `https://www.azsecure-data.org/`, which is an open source data portal specialising in security research (Samtani et al., 2016). Whereas D2, with 11,832 posted messages, was first used in Queiroz et al.

(2017) and can be found at `http://tiny.cc/8ws67y`. In all datasets, the content of the posted messages represents technical and personal communication regarding computing, security, internet services, and technology. The users of these forums range from people interested in computer security subjects to sellers and buyers of hacker products. Moreover, among the messages, we identify several posts relating to private data exposure (such as credit numbers and user accounts), and sharing of copyright software (activation number and cracked software). These activities are considered a by-product of a hacker attack, which have not mentioned whether it was performed through a software-security flaw, thus, are not part of what we consider as target messages in this research.

After a thorough inspection of these datasets, it was noticed that messages that represent the goal of this research are found in a small number, i.e., posts with information about vulnerabilities in software (exploited or not). This situation left us with an imbalanced number of instances; fewer from target class compared to non-target, which is not ideal for training classification models (Ganganwar, 2012). Taking this into account, in Section 3.3, we describe the steps taken to sample datasets and to enhance the balance between the target and non-target messages.

## 3.2 Datasets Description

This section describes the principal activities of the dataset sources used in this work, for instance, the description of its purpose, content and number of users.

#### D1 - CrackingArena Forum

This is one of the largest hacker forums existing in 2018 with 44,752 posts and 11,977 active users from February of 2013 to February of 2018. It contains security communication regarding issues in computing, which makes these messages suitable to study the interaction among hackers. The variety of topics in the forum covers areas such as social engineering, cracking/exploit tools and tutorials. Also, this forum is known as one of the principal sources for learning how to hack newly emerged problems in computer systems. The data was collected by Samtani et al. (2016).

**D2 - Security Experts**

The data was collected from 12 security-expert users on Twitter by Queiroz et al. (2017). There are two groups of professionals in this dataset: The well-known security experts with an average number of followers of 18,800, and lesser-known security experts, with an average number of followers of 1,100 between 2016 and 2017. Each group contains 6 users. Their posts are mostly related to security aspects of technology, including software vulnerabilities and hacking. The collected Tweets range from early November of 2015 to early March of 2017. The total number of Tweets gathered is 11,833.

**D3 - Dream Market**

The Dream Market is considered the largest marketplace on the Dark Web space after the shutdown of Alphabay forum (CNET, 2017). With 91,463 posted products from 2,092 sellers in 2016, this is a well-known marketplace for selling illegal products, such as illicit drugs, fake IDs, stolen credit card numbers and copyrighted software. It is also a place for advertises hacker products and services used in malicious hacker activities. Some of these products are generally associated with cyber-attacks on companies and governmental agencies. This marketplace is within the Dark Web domain, which is only accessible by using a special communications setup called a ToR network. This network is used to provide buyers and sellers with privacy and anonymity. The data was collected by Samtani et al. (2016)

**D4 - Garage4Hackers Forum**

This is a medium-sized forum in terms of the number of messages, with only 8,699 messages that span from June of 2010 to September 2017. The messages in this forum are related to exploitation, botnets and explanation of reverse engineering in software. It also has information regarding specialised hacking tools for the *reconnaissance* phase of cyberattacks. The data was collected by Samtani et al. (2016)

**D5 - Cracking Fire Forum**

This forum has approximately 14,511 active users and 36,684 messages from April 2011 to February 2018. Additionally, after brief inspection, we noticed that some of the posts contained pieces of pseudo-code and source code, which are aimed to perform malicious operations, such as compromise online social media accounts. Thus, this dataset facilitates cyber security research on software/web applications vulnerability exploitation tools. The data was collected by Samtani et al. (2016)

## 3.3 Dataset Preparation

Our approach is based on supervised classification, which needs training data with suitable labels. However, the original datasets lack the provision of these labels for the messages. Additionally, based on organised inspection of the contents, we have noticed that the target messages are found in a low proportion compared to the non-target messages. As a result, to solve this problem, we propose a robust systematic procedure for sampling and labelling these datasets.

For performing the labelling preparation of the datasets, the dataset should follow three principal requirements:

- Each message is identified as belonging to either a *target* or *non-target* class.

- Each message needs to be correctly assigned to a label, otherwise the performance of the model would be negatively affected, which would result in failure in detecting the software vulnerability communication.
- There should be, ideally, a balanced number of instances in each class to provide enough information for pattern learning.

The labelling task is at the core of supervised learning approaches as the dataset needs to be previously labelled before the creation of supervised models. In some cases, this task requires expert human knowledge, which is time-consuming and expensive depending on the size of the dataset. Due to the lack of labelled datasets (gold standard) in security research for building these models, researchers are either labelling their own datasets or (2) using automated services (e.g., Amazon Mechanical Turk crowdsourcing). Considering this problem, the choice in this thesis was performing our own labelling scheme using expert knowledge, as these messages carry out a degree of ambiguity, which makes it difficult for non-experts to determine its real category. As seen in Table 3.2, a certain background knowledge is needed, to clearly understand the content.

As seen MSG-1, marked as Yes, this message is related to a type of vulnerability (Stack Buffer Overflow) affecting a software product. MSG-2, also marked as Yes, is related to a release of a Proof Of Concept (PoC) of a vulnerability called dirtycow. The posts MSG-3 and MSG-4 are related to personal opinion and have no direct relation to real vulnerabilities in software. Despite MSG-3 and MSG-4 having hack and hacker keywords, they are not related to the distribution of messages about software vulnerability communication and are thus marked as No.

Moreover, in MSG-5, there is not enough information to decide whether either the ssh scan tool is vulnerable or can be used against vulnerable software. Likewise in MSG-6, we cannot confirm that the error mentioned leads to a vulnerability into the sneaker software product, thus they are marked as Undecided. The label Undecided is used to capture any subjectivity doubt of the labeller regarding the real class of the message, target (Yes) or non-target (No). Further details on the labeling task is given in Section 3.3.2.

We acknowledge that the model will only be as good as the knowledge of the labellers, when it comes to detecting hacker posts. For this reason, labellers who understand the ambiguity and subtlety of the posts are a critical component in our work.

Table 3.2: Message Examples

| *ID* | *Message* | *Label* |
|---|---|---|
| **MSG-1** | Multiple remote memory corruption vulns in all Symantec/ Norton antivirus products, including stack buffer overflows | Yes |
| **MSG-2** | PoC for dirtycow vuln [URL] | Yes |
| **MSG-3** | Reading about lawyers argue about our Jeep hack is endless fun | No |
| **MSG-4** | it is amazing a hacker can put up with a sociologist ;) | No |
| **MSG-5** | Just released ssh_scan v0.0.10. Release notes can be found here | Undecided |
| **MSG-6** | I like sneaker's error 0xC0000156 | Undecided |

In other words, to guarantee the quality, we decide to not outsource the task to non-experts nor use automated techniques for labelling the messages. In the following sections, we describe the detailed steps for preparing the dataset to comply with the requirements. The steps are: (1) Keywords-filtering; (2) Labelling Annotation; and (3) Final Label Assignment.

### 3.3.1 Keyword-filtering

The original dataset has brought the following issues: (1) there is a high number of messages to be manually labelled by human experts (193,430 in total), and (2) there are few instances of the target class, which is found in less proportion compared to the non-target class.

Perform the labelling in all messages is infeasible due to our limited number of expert knowledge to perform the task. Also, reducing the number of messages to be labelled by eliminating the non-target messages (majority) while keeping the target class (minority) contributes for balancing the dataset.

For this reason, we have followed similar approach as seen in Chen et al. (2017b), König and Brill (2006) for selecting a sample of each dataset. These approaches have defined keywords and patterns to select a representative sample of messages of the datasets. Then, the selected data is assessed by a human expert to define the real class. In summary, this procedure aims to:

1. Reduce the number of messages to be labelled by expert labellers, thus, reducing the time and human resources needed for completing the task.
2. Increase the number of messages that represents the target class (software vulnerability communication) in the reduced sample.

To achieve these goals, we used a set keywords from a *specialised keywords list* related to software vulnerabilities. This procedure allows us to reduce the amount of original data while increasing the chances of including the target class messages in our final dataset. After manual evaluation, we have noticed that we had increased the proportion of target class instances compared to non-target.

### Filtering using Specialised Keyword List

Chen et al. (2017b) has used the keywords for reducing the amount of data to be labelled for an abusive comments detection task. Similarly, we have used a security keyword list to extract a set of sample messages form our original datasets. This list contains terms used to describe common security problems in software, for instance, Cross-Site Scripting (XSS), buffer overflow and SQL Injection (SQLi), which are commonly used to communicate software security vulnerabilities.

These terms are provided by two well-known list software defects: The Open Web Application Security Project (OWASP) top 10 Application Security risks (OWASP, 2017) and the SANS top 25 software errors (SANS, 2019). The full list can be seen in Appendix A1.

In Table 3.3, we present the number of messages that contain at least one word from the list as well as messages without any of these keywords for each dataset. All duplicated messages are excluded. The reduction of messages compared to the original dataset is 55% , from 193,430 to 88,359 instances. The number of messages containing at least one keyword is 34,145, and there are 54,214 messages that not contain any keywords.

Table 3.3: Messages with and without Word from The List

| *ID* | *No. of inst.* ***with*** *keyword* | *No. of inst.* ***without*** *keyword* | *Sum* |
|---|---|---|---|
| **D1** | 1,353 | 16,421 | 17,774 |
| **D2** | 6,173 | 5,659 | 11,832 |
| **D3** | 18,881 | 7,877 | 26,758 |
| **D4** | 3,972 | 4,659 | 8,631 |
| **D5** | 3,587 | 19,598 | 23,185 |
| **TOTAL** | 33,966 | 54,214 | 88,180 |

**Sampling Messages**

Performing the labelling task on 88,180 messages as shown in Table 3.3 remains time-consuming. To solve this problem, we sampled the messages of each dataset by reducing the total of messages to be labelled to 10,000, according to the following:

- Random selection of 1600 messages that contain **at least one** security keyword (list) from datasets (D1, D2, D3, D4, D5)
- Random selection of 400 messages **without** security keywords from datasets (D1, D2, D3, D4, D5)

Additionally, the messages are presented in a range of different sizes regarding the number of words. Capturing a representative sample of these sizes is relevant to the training phase of the model, as the model would learn different examples, thus improving prediction performance. As a result, to capture a representative sample, we have performed the following three steps:

1. Exclude the upper and lower outliers messages in terms of the number of words.
2. Divide each dataset into 4 different groups (A, B, C, D) according to its number of words distribution, where group A represents all messages up to the first quartile (Q1), B represents messages from Q1 to the second quartile (Q2), C represents messages from Q2 to the third quartile (Q3) and D represents the messages from Q3 onward. An illustrative example is seen in Figure 3.1.
3. From messages that contain at least one of the specialised keywords, we randomly select 400 messages per group and per dataset, whereas from messages without any keywords, we randomly select 100 messages per group and per dataset.

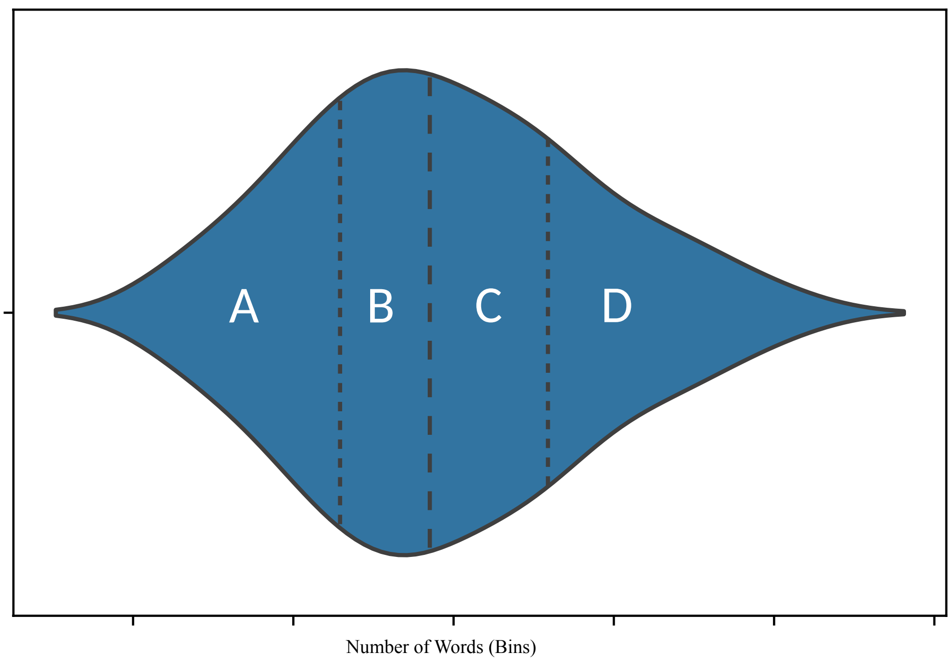


Figure 3.1: Illustrative Example of Groups A, B, C, D Divided by Interquartiles Q1, Q2, Q3 (Dotted Line)

Finally, we provide the violin graphs in Figure 3.2 of the sampled (reduced) dataset. This figure is a graphical representation of the distribution of messages per number of words in each dataset. The three dotted lines represent the first (1Q), second (2Q) and third (3Q) quartile. Comparing the dataset, we see that the majority of the data is spread within the 0 to 350 words range, only D3 has a large spread, of 0 to 650 words.

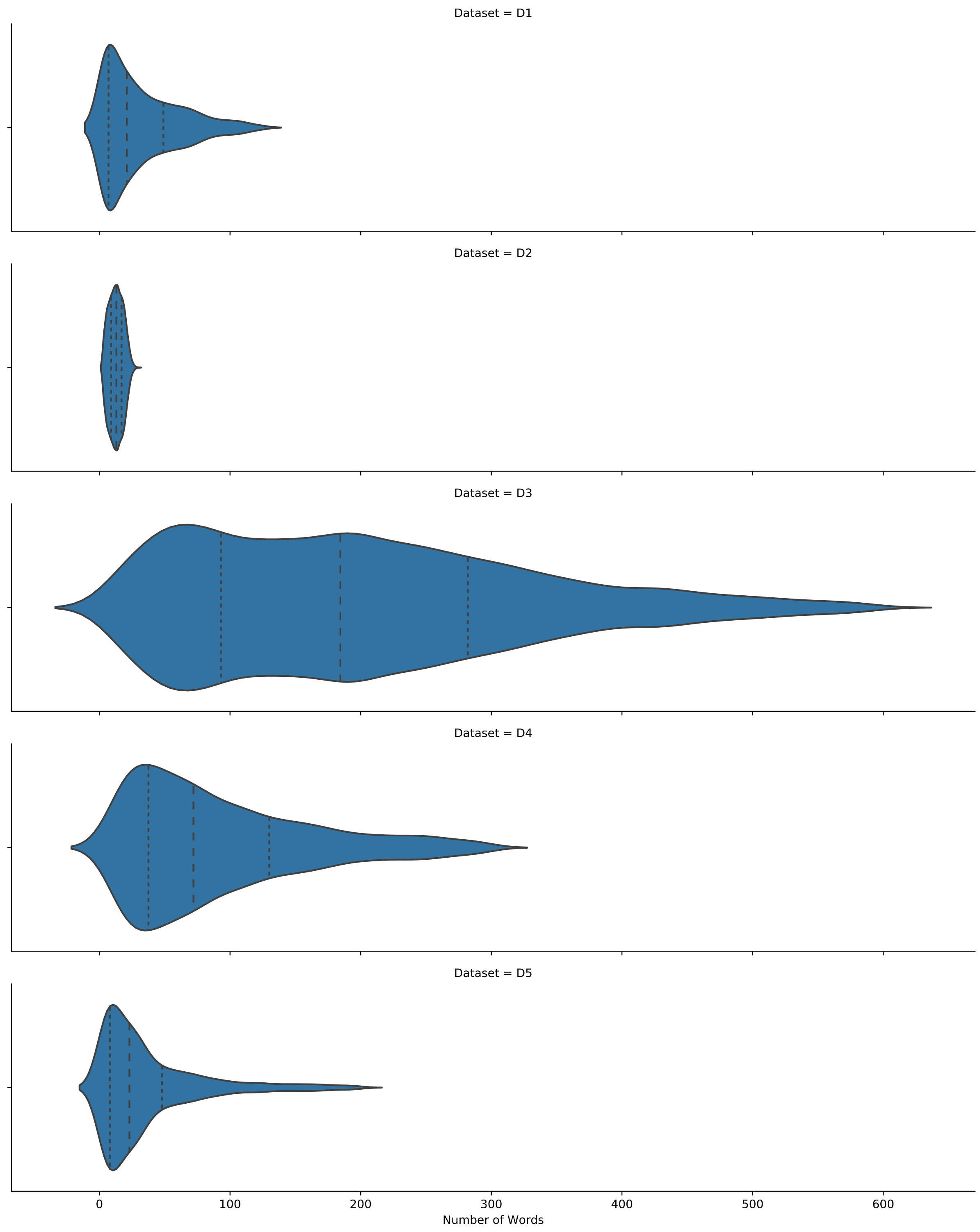


Figure 3.2: Distribution of Messages per Number of Words

### 3.3.2 Label Annotation Task

Accurate labels are critical to the success of supervised learning. As previously mentioned in Chapter 2, there are two principal approaches for acquiring dataset labels: Expert and Crowdsourcing. In our 5 datasets, we have used the expert labelling approach, as non-experts might have difficulty understanding the content of the messages. Furthermore, to guarantee high quality of labels, we have performed the labelling task with multiple annotators.

To achieve our goal, we followed a systematic approach, where each instance (post) in the datasets has been labelled by three different human labellers (computing researchers). Those three opinions are considered for use in an additional step for defining the final label (the majority votes). The use of three experts brings more expertise to the labelling process, whereas the majority of votes approach reduces their influence (subjectivity) on the final result. Additionally, to facilitate the decision of labellers, we created a third label option called **Undecided**, which can be used for ambiguous massages where the labellers are not sure whether it belongs to target or non-target class.

In order to perform the task, the labellers were asked to consider the following rules when deciding whether the message is related to software-vulnerability-related communication:

- Yes, for messages that appear as malicious messages regarding how to breach vulnerabilities in software assets.
- No, for messages not related to hacker activity or which are out of the scope of our research (Data breach, copyrighted software cracked, stolen accounts and credit card accounts).
- Undecided, for messages that the labeller does not have enough information or confidence to mark as Yes or No.

Existing approaches, as in Nunes et al. (2016), have relied on labelling annotation done by the authors, where they provided a discussion on doubtful messages to form a consensus regarding the final label. In Deliu et al. (2018), the authors have assigned labels to specific instances in a keywords-matching approach, where messages containing specific keywords, e.g., "Hacker", are marked as being from target class. In the former approach, the authors have considered that some messages can be difficult to label due to the ambiguity of certain posts, whereas in the latter approach, the authors have ignored the ambiguity problem by assuming the keywords matching would suffice to determine which class a post would belong to. We acknowledge that the last approach is not suitable for this research, as we see in our dataset, we have security-specific keywords appearing in target and non-target messages, e.g., "hacker" or "vulnerability".

### On-line Survey for Labelling Task

The on-line survey was the method used to collect the labels. This survey consists of the following three parts:

1. An introductory text regarding the dataset to be labelled (Appendix A2),
2. A question in which the answer should be **Yes**, **No** or **Undecided**
3. Rules and Tips for marking the labels.

For part (2), the question to each dataset was set into the domain of software vulnerabilities communication, however, for D3 (marketplace), it was adjusted to support the content of each this specific dataset. Differently from D1, D2, D4 and D5, the dataset D3 was collected from a marketplace forum, where the principal activity is the trade of malicious and criminal products, which includes malicious software. Furthermore, the questions can be seen in Table 3.4. we highlight that the term SOFTWARE in these questions is used in a broad context, which represents every piece of software running on computers, such as desktop computers, web servers, ATMs, embedded systems, mobile phones, as well as network protocols. We have also provided examples of target and non-target messages per dataset in Table 3.5. The entire dataset with all messages can be found at `http://tiny.cc/8ws67y`.

Table 3.4: Questions on Survey

| *Dataset* | *Question* |
|---|---|
| **D1** | Is this message somehow related to exploitation of software vulnerabilities? |
| **D2** | Is this message somehow related to exploitation of software vulnerabilities? |
| **D3** | Can this product/service be used for malicious exploitation of software vulnerabilities? |
| **D4** | Is this message somehow related to exploitation of software vulnerabilities? |
| **D5** | Is this message somehow related to exploitation of software vulnerabilities? |

Table 3.5: Examples of Target and Non-target Messages per Dataset. Yes = Target, No = Non-target

| *DATASET* | *MESSAGE* | *LABEL* |
|---|---|---|
| **D1** | Hello everybody, First , i am a big newbie so please don't blame me ! I want to crack a software in .Net So first i scan my exe with protection ID : Everything is good exe not protectected I run .Net reflector 8.3 and i see strange line, i run De4Dot ! I relaunch Net reflector and use search system (Tools - Search or F3) i type. serial, Verification, Licence , key ,validate , login , password ... But no results I search in the program and look for things that look like a function that requires a serial ! I found : License , but i don't know where to search ? Can you help me please ? Thanks a lot for help! | Yes |
| | Our Free VPN (Virtual Private Network) server is Designed with the latest technologies and most advanced cryptographic techniques to keep you safe on the internet from prying eyes and hackers by securely routing all your internet traffic through an encrypted tunnel to bypass government censorship, defeat corporate surveillance and monitoring by your ISP. VPNBook strives to keep the internet a safe and free place by providing free and secure PPTP and OpenVPN service access for everyone. You do not have sufficient rights to see the hidden data contained here. | No |
| **D2** | [RT] [USERNAME] Signal bug lets attackers tamper with encrypted messages patch now [URL] | Yes |
| | Last talk, android firewall by [USERNAME]. Not really into android but tweeted out of respect for [USERNAME]. [URL] | No |
| **D3** | Droid Jack - Android RAT 4.4 -Droid Jack - Android Rat 4.4 + Guide There is nothing that you can do with a PC that you can't do using an Android phone. Since the power in the hand has grown so much, a control over that power is also needed. DroidJack is what you need for that. DroidJack gives you the power to establish control over your beloveds' Android devices with an easy to use GUI and all the features you need to monitor them. droid jack, android rat, droidjack, rat, trojan, android hack, hack, droid hack, android spy, malware, spyware If you have any question, let me know. droid jack, android rat, droidjack, rat, trojan, android hack, hack, droid hack, android spy, malware, spyware | Yes |
| | Homemade Grenade Launchers Let Uncle Ragnar walk you through these simple step-by-step plans for making an M79 or M203 in your own workshop! All it takes is ordinary tools and some pipe, washers, nuts and bolts . Reloading info for 40mm ammo and BATF guidelines are included | No |
| **D4** | suppose a programmer wrote: Code: touch ("myfile");chmod ("myfile",700); instead of: Code: umask (077) ;touch ("myfile"); race condition can be exploited by opening the file for reading right after the touch command was executed and before the chmod command took place. | Yes |
| | Adobe has introduced technology that makes it easier for users to delete local shared objects (LSOs), known as Flash cookies LSOs store user preferences, but some websites have been using the LSOs to restore user cookies even after users have manually deleted them. Working with Mozilla, Google and Apple, Adobe has developed an application programming interface (API) known as NPAPI ClearSiteData that lets users delete LSOs from the settings panels of certain browsers. For more info: <br> [URL] | No |
| **D5** | The ultimate tool used for spying your desired contact who's using WhatsApp messenger to speak with friends, partners and family with mobile devices (Androind, iOS, BlackBerry, Windows phone, Nokia). Our tool is not just made for spying purposes, but there are also options to type and send messages from your desired target's WhatsApp account to someone of their contacts. You can also update their status message, and many more! Choose any phone number you want,from any country in the world and hack any desired WhatsApp account in just few minutes! [URL] | Yes |
| | its ok virus no mean to be mad just saying not like used to be .. i had others just deleted post .. u are great just people change pws and by paying every month i only look for a few it get frustrating sorry | No |

### 3.3.3 Final Label Assignment

For performing the Final Label task, we have used the labels from the labelling annotation task, where each message was annotated by multiple labellers (3 different expert labellers). According to Sheng et al. (2008), this approach is a simple but effective alternative for labelling noisy and ambiguous instances, which provide better quality labels than those provided by a single labeller. As a result, we have provided labels using two different schemes: The **Unanimous Label Assignment (ULA)** scheme, where the final label is given only when all 3 labellers have agreed on the label; and the **Partial Label Assignment (PLA)** scheme, where the final label is given when at least 2 of 3 labellers have agreed on the label.

Some examples of Final Label (FL) using **ULA** scheme are given as follows:

- FL = $NO$, if $L_a = NO$, $L_b = NO$, $L_c = NO$, with $n \in \{a, b, c\}$ being a different labeller ($L$).
- FL = $YES$, if $L_a = YES$, $L_b = YES$, $L_c = YES$, with $n \in \{a, b, c\}$ being a different labeller ($L$).
- FL = $UNDECIDED$, if $L_a = UNDECIDED$, $L_b = UNDECIDED$, $L_c = UNDECIDED$, with $n \in \{a, b, c\}$ being a different labeller ($L$).

Some examples of Final Label using **PLA** scheme are given as follows:

- FL = $NO$, if $L_a = NO$, $L_b = NO$, $L_c = UNDECIDED$, with $n \in \{a, b, c\}$ being a different labeller ($L$).
- FL = $YES$, if $L_a = YES$, $L_b = YES$, $L_c = UNDECIDED$, with $n \in \{a, b, c\}$ being a different labeller ($L$).
- FL = $UNDECIDED$, if $L_a = UNDECIDED$, $L_b = UNDECIDED$, $L_c = NO$, with $n \in \{a, b, c\}$ being a different labeller ($L$).

Additionally, for PLA scheme, messages with unanimous agreement are also included, e.g.: $L_a = \mathit{YES}$, $L_b = \mathit{YES}$, $L_c = \mathit{YES}$ or $L_a = \mathit{NO}$, $L_b = \mathit{NO}$, $L_c = \mathit{NO}$ or $L_a = \mathit{UNDECIDED}$, $L_b = \mathit{UNDECIDED}$, $L_c = \mathit{UNDECIDED}$. Furthermore, we have excluded all total disagreement messages, e.g.: $L_a = \mathit{YES}$, $L_b = \mathit{NO}$, $L_c = \mathit{UNDECIDED}$ before performing PLA and ULA. In Table 3.6, we present the number of instances, percentage and remainder quantity of instances excluded from the final dataset.

Table 3.6: Total Disagreement

| *ID* | *Excluded* | *% Exclusion* | Remainder |
|---|---|---|---|
| **D1** | 71 of 1,753 | 4% | 1,682 |
| **D2** | 73 of 2,000 | 3% | 1,927 |
| **D3** | 79 of 2,000 | 4% | 1,921 |
| **D4** | 44 of 2,000 | 2% | 1,966 |
| **D5** | 26 of 2,000 | 1% | 1,974 |

Moreover, we can see the summary description of the ULA scheme (Table 3.7) and PLA scheme (Table 3.8). It is seen that, by using ULA scheme, 25% of the instances (on average) are excluded, whereas by using PLA scheme, the exclusion of instances is around 3%, on average, being only those with full disagreement. As a consequence, in order to maintain a large number of instances within the final dataset, we decided to use the PLA scheme for the remainder of this work.

Table 3.7: Unanimous Label Agreement Scheme Description

| *ID* | *Scheme* | *No. of No* | *No. of Yes* | *No. of Undec* | *% Distrib. (no/yes/und)* | *Total* | *Excluded / (%)* |
|---|---|---|---|---|---|---|---|
| **D1** | ULA | 1,243 | 27 | 7 | 97/3/<1% | 1,277 of 1,682 | (405)/24% |
| **D2** | ULA | 1,372 | 110 | 0 | 93/7/0% | 1,482 of 1,927 | (445)/23% |
| **D3** | ULA | 1,365 | 89 | 3 | 94/6/<1% | 1,457 of 1,921 | (463)/32% |
| **D4** | ULA | 1,307 | 70 | 0 | 95/5/0% | 1,377 of 1,966 | (589)/30% |
| **D5** | ULA | 1,525 | 15 | 0 | 99/1/0% | 1,540 of 1,974 | (434)/22% |

Table 3.8: Partial Label Agreement Scheme Description

| *ID* | *Scheme* | *No. of No* | *No. of Yes* | *No. of Undec* | *% Distrib. (no/yes/und)* | *Total* | *Excluded / (%)* |
|---|---|---|---|---|---|---|---|
| **D1** | PLA | 1,520 | 114 | 48 | 90/7/3% | 1,682 of 1,682 | (0)/0% |
| **D2** | PLA | 1,633 | 243 | 51 | 85/13/2% | 1,927 of 1,927 | (0)/0% |
| **D3** | PLA | 1,638 | 211 | 72 | 85/11/4% | 1,921 of 1,921 | (0)/0% |
| **D4** | PLA | 1,704 | 225 | 37 | 87/10/3% | 1,966 of 1,966 | (0)/0% |
| **D5** | PLA | 1,864 | 110 | 0 | 95/5/0% | 1,974 of 1,974 | (0)/0% |

### 3.3.4 Time Span of Final Dataset

After labelling the final dataset with the PLA scheme, we ended up with instances spread across a range of months and years. Fig 3.3 presents the volume of messages in each dataset over time, where the x-axis represents the year-month of the post, while the y-axis represents the volume of posts on that period. These messages are spread over a period between June of 2010 and February of 2018 with each dataset covering a subset of this range of time. The dataset D1 contains messages between 2013-02 and 2018-02 (5 years), D2, the shortest dataset in terms of temporal range, contains messages between 2015-11 and 2017-03 ($\approx$ 1 year). D4, the longest, contains data between 2010-06 and 2017-09 ($\approx$ 7 year), D5 between 2011-04 and 2018-02 ($\approx$ 6 year). Only D3 is not present in this figure as the publishers in Samtani et al. (2016) have not provided the timestamp of the messages. The messages and their timestamps are used to perform a deployment simulation and Concept Drift experiment presented in Chapter 6.

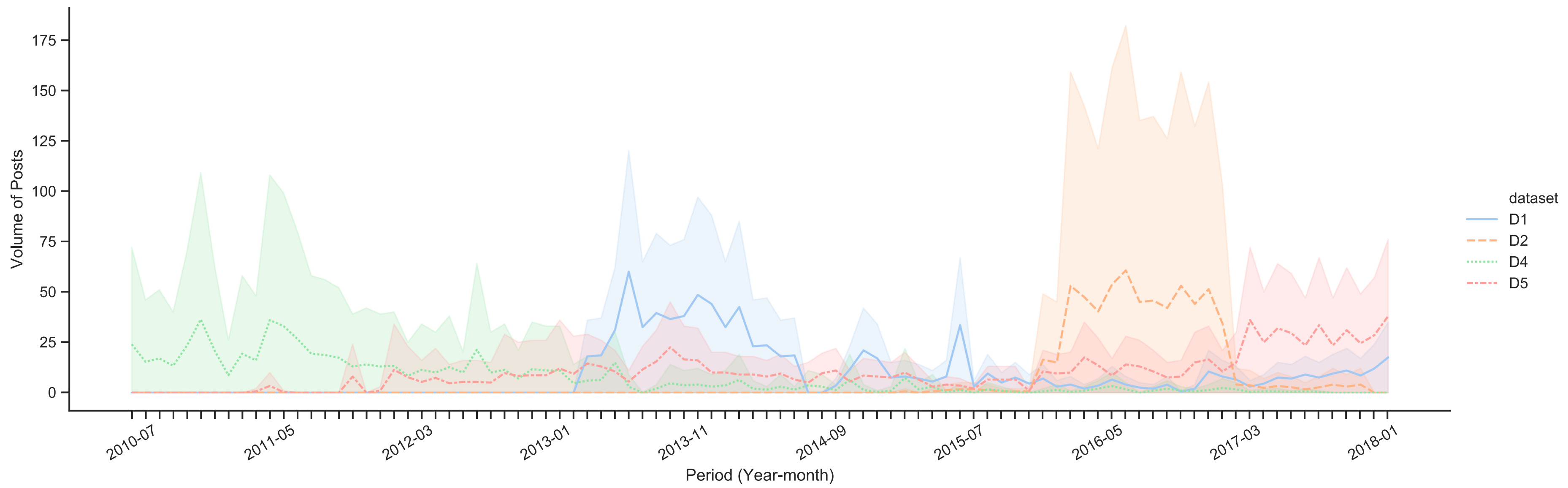


Figure 3.3: Volume of Post and Overlapping Through Time

## 3.4 Methodology

This section aims to briefly introduce our choice of algorithms and feature representations used to build the models investigated in this thesis. We have based our choice on techniques that have shown promise in the text mining domain and previously discussed in Literature Review Chapter 2. The algorithms are presented in two different categories, Traditional and Deep Learning-based. Likewise, the feature representations are divided into Traditional and Distributed representation. Furthermore, we describe the evaluation design and metrics used to perform the analysis of the different models built with 5 different datasets.

### 3.4.1 Supervised Classification Approaches

Supervised approaches can be categorised as *regression*, when the outcome value (output) provided is continuous (numerical), or *classification*, when the output is categorical value (or classes). In this work, the focus is on classification models, thus, we are using malicious (target) and non-malicious (non-target) communication as categories for the output values.

Additionally, this approach requires the mapping of each input instance of the dataset to its output value, in a process known as *labelling*. That is the principal characteristic that differs from *unsupervised* approaches, which require labelled instances.

Although this work is primarily using supervised methods for classification, we also use unsupervised approaches (language models) for feature representations (as seen in Section 3.4.2). The motivation for using supervised approaches is based on the following:

- It allows us to create a model that learns with previous data without the need of predefining rules, which differs from expert systems (Fogel et al., 1993), and
- It allows us to objectively evaluate the performance the models with quantita-

tive metrics. This differs from unsupervised methods, which lack objective metrics for measuring the performance,relying more on qualitative analysis (Palacio-Niño and Berzal, 2019).

In this thesis, we have selected a range of supervised algorithms, which include traditional and state-of-the-art deep learning classification algorithms. Traditional algorithms can provide comparable results with deep learning algorithms in certain tasks, whilst being more computationally efficient and require less training data (Chen et al., 2019, Deliu et al., 2017, Kim, 2014). Although the foremost goal of these algorithms is providing the best accuracy for the models, the computational performance and efficiency are also important characteristics that should be taken into consideration, especially when the intention is to deploy these models for real-life use. For this reason, we have decided not only to investigate the state-of-the-art algorithms, but also the performance of traditional algorithms in our specific task. As a result, the experimental procedure in Chapters 4, 5 and 6 is the evaluation of the most commonly used traditional algorithms as well as the state-of-the-art Deep learning-based algorithms for text classification as following:

#### Traditional Algorithms

Traditional algorithms usually rely on hand-crafted features to provide the final prediction. In text classification domain, the most used are Support Vector Machine (Cortes and Vapnik, 1995) and Naïve Bayes (Lewis, 1998, Sebastiani, 2002). More details are described In Chapter 4.

#### Deep learning-based Algorithms

Deep Learning algorithms are based on Artificial Neural Network architecture. This architecture finds itself the best set of features within the input data. Recently, Convolutional Neural Network (CNN) (Kim, 2014, Lecun et al., 1998) and Recurrent Neural Network (RNN) are the most used architectures, with the former being adapted for text classification, and the latter being used mostly in interdependent sequential data, for instance, sound and text. More details are described in Chapter 5.

### 3.4.2 Feature Representations

This work has focused on using only textual features, such as words, characters and sentences. These parts provide the meaning to messages and are also ubiquitous to all hacker platforms and social media used to disclose text-based information. For this reason, we have selected a diverse set of feature representations that provide different forms of representing the characteristic of the message, for instance, some representations are focused on the importance of frequent keywords in the context and others are focused on aligning similar words and sentences together. Therefore, the principal goal is to evaluate these feature representations in a downstream classification task, in other words, to find the representation that provides best accuracy in combination with classification models for detection software vulnerability communication.

We acknowledge that adding different types of features (crafted features) might provide an enhancement of performance of the model. However, some of them might be available to specific datasets and not the other. For instance, the use of re-tweets (RT) as done by Sabottke et al. (2015) is useful only for Twitter data (D2), as the other datasets (D1, D3, D4 and D5) do not have re-tweet-like function implemented in their platforms (hacker forums). The same occurs for hacker-rating attributes used by Benjamin and Chen (2012), which are available in some hacker forums (D1, D4, D5), however, it is not present in Twitter data (D2).

As a result, by using only word features, we can compare the results of the models fairly and also evaluate the importance of each language model for downstream classification tasks. Furthermore, in this thesis, we have divided the Feature Representations into two categories as done in Chen et al. (2019), the *Traditional* and *Distributed* Representation, in which the latter includes the Word and Sentence Level representations, as described below:

### Traditional Representation

*Bag-of-words* (BoW), *word n-grams* and *char n-grams* are commonly used in several text-mining and classification tasks Benjamin et al. (2015), Lee and Kang (2019), Nunes et al. (2016). With BoW, the sentence (in our case, the post messages) is split into a set of tokens (words), then each token is counted to produce a vector that represents the entire sentence. As the previous representation, *words n-grams* and *char n-grams* are also token-based, however, the number of word/characters representing tokens is defined by *n*, where $n \geq 1$. The main difference between *bag-of-words* and *word n-grams* models is that the latter encodes contextual information in a sequence of words when $n > 1$, i.e: The words "United" and "States" when appearing in sequence might illustrate a different context compared to appearing separately. Finally, the *char n-grams* representation is better for representing sentences with rare words and morphological variants. Some of the common characteristics of these language models are:

- It forms a sparse vector (with lots of zeros)
- The word order of messages is lost

### Distributed Representation

As mentioned in Chapter 2, these categories of language models can be subdivided into two other categories, the *Word Level Distributed* representation and *Sentence Level Distributed* representation. In this thesis, we have selected models from both categories, as they have different characteristics. Examples of the former category are *Word2vec* and *Glove*, and the latter are *Sent2vec*, *InferSent*, and *SentEnconder*. We have used these models in Chapter 4 and 5, their details are given in the respective chapters. In following, we summarise their common characteristics and improvements over traditional representation:

- It forms a dense vector
- Semantically and Syntactically similar words (or sentence) are mapped together

- In case of Sentence Embedding, the order of messages is maintained

### Comparison of Traditional and Distributed Representation

In order to investigate the optimal feature representations that enhance vulnerability communication detection, we have selected language models with different characteristics, which are summarised in Table 3.9. It is seen that, the traditional is characterised for being sparse, while Word and Sentence distributed representation are Dense Vector. The sparsity of the vector might present a well-known issue called "curse of dimensionality" that affects the performance of the model (Bellman et al., 1957). However, this problem is not presented in dense representation, therefore is more computationally expensive than traditional representations. In terms of providing similar words/sentences together in a vector space, both Distributed representation models present this property. However, only the Sentence Distributed representation distinguishes between sentences that use similar words but different word order, i.e, "Play hard, don't study! and "Study hard, don't play!" would be considered as equal sentences in Traditional and Word Level distributed representations (Deepanshu Jindal, 2019).

Table 3.9: Summary of Language Models

| | **Sparse Vector** | **Dense Vector** | **Similarity** | **Word Order** |
|---|---|---|---|---|
| TRADITIONAL REPRESENTATION | | | | |
| **Bag-of-Words** | X | | | |
| **Word n-gram** | X | | | |
| **Char n-gram** | X | | | |
| WORD LEVEL DISTRIBUTED REPRESENTATION | | | | |
| **Word2vec** | | X | X | |
| **Glove** | | X | X | |
| SENTENCE LEVEL DISTRIBUTED REPRESENTATION | | | | |
| **Sent2vec** | | X | X | X |
| **InferSent** | | X | X | X |
| **SentEncoder** | | X | X | X |

### 3.4.3 Evaluation and Metrics

As the principal goal of the models is to detect the messages related to vulnerabilities in software among those non-related, we acknowledge that the impact of a false negative (FN) classification (assigning a true malicious communication as being non-malicious) is more damaging than the impact of a false positive (FP) (assigning a false malicious communication as being malicious) in real-life application of these models. As a consequence, when a model achieves a high rate of FN, we miss the identification of the target messages, thus, we lose the chance of using this information for proactive actions against likely cyber events.

On the other hand, a model with high rates of FP is not desirable either, as it implies that a model is wrongly detecting the non-target as target messages. If this situation occurs frequently, either a time-consuming expert investigation will be needed, or unnecessary security actions will have to be taken.

Under these circumstances, the metrics used to evaluate the models need to consider the real-life application of these models. Therefore, we have chosen the *Positive Recall* and *Average Class Accuracy* (or average recall) as metrics for evaluate the models. The majority of works in this area has focused on recall (Lippmann et al., 2016, Mulwad et al., 2011, Nunes et al., 2016, Sabottke et al., 2015, Trabelsi et al., 2015). However, we understand that the recall metric does provide the full picture for deployment in real-life situation, as this metric does not inform the true negative assigned as positive.

The Positive Recall (1), is given by TP, which is the total of instances correctly classified as positive (malicious communication), over TP+FN, which is the total number of instances from that class. Whereas the Average Class Accuracy (2), is given by the sum of the recall of positive class and the recall of the negative class divided by the number of classes (2 classes, in this case). This metric is also suitable for the evaluation of models built over imbalanced datasets as it prevents the majority class from dominating the results and presents a balanced result over True and False positive detection. These metrics are also present in similar work using imbalanced datasets (Brodersen et al., 2010, Chen et al., 2017a, Urbanowicz and Moore, 2015).

These metrics are used over 10-fold cross-validation evaluation in the following Chapters 4, 5 and 6.

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{1}$$

$$Avg.ClassAcc = \frac{Recall(pos.Class) + Recall(neg.Class)}{No.Classes} \tag{2}$$

## 3.5 Summary

This chapter describes the datasets that will be used in the experiments throughout this thesis. These datasets were collected from 5 different sources and 3 distinct Internet domains (Surface, Deep and Dark web). Due to the size of these datasets, we propose a structured scheme for reducing the number of messages, thus providing a feasible dataset size for the labelling task. As we have observed in the literature review, there is a lack of gold standard dataset in security domain research. For this reason, we have performed a structured labelling task using multiples annotators to guarantee that the final labels are not biased by the subjectivity and background knowledge of the labellers.

Furthermore, we describe the methodology used to create and evaluate the classification models. This methodology purposes an evaluation of a range of state-of-the-art language models and classification algorithms to detect software vulnerability communication using multiple datasets. For the language models we have included the traditional *bag-of-words, and word/char n-grams*, as well as *Word and Sentence Embedding*; for algorithms, we have selected the traditional Support Vector Machines (SVM) and Naïve Bayes, and robust Deep Learning architectures, for instance, CNNs- and RNN-based algorithms.

Finally, in all experiments, we have used Python 3.6.0v programming language, Sci-kit 0.20.2v and Keras API for TensorFlow 1.15.0v as libraries.

CHAPTER 4

# Detection of Software Vulnerability-related Communication Using Traditional Text Classification Models

In the age of information, criminals are taking advantage of social media channels to share hacking tools and promote coordinated cyber-attacks (Algarni and Malaiya, 2014, Deliu et al., 2018). In this context, cyber security analysts are striving to find new approaches to extract usable information from these sources that helps to counter malicious attacks against enterprise systems.

As a consequence, an emerging approach nowadays is the investigation of security-related messages exchanged on social media channels from the most variate domains, for instance, forums and marketplaces found in Deep/Dark Web, and Twitter in Surface Web channels (Lippmann et al., 2016, Nunes et al., 2016, Sabottke et al., 2015).

In this chapter, we conduct a range of experiments using traditional machine learning algorithms and word features representation to create classification models that detect software-vulnerability communication in social media posts. These experiments include an analysis of classification models through a variety of techniques, such as features extraction/selection, and binary/multi-class classification. To complete this task, we systematically evaluate and compare the performance of these models. These experiments were conducted by using three different datasets, each one coming from a different source (Forums, marketplace and Twitter). Additionally, due to the problem of imbalanced data mentioned in Chapter 2, we investigate sampling techniques that improves the detection accuracy of these models.

The principal goal is to perform a thorough, empirical investigation of these models to identify the optimal techniques that better suit our purpose, the detection of software-vulnerability communication in social media. We believe that this study will support the use of machine learning models in Cyber Security Intelligence (CTI).

The following sections are organised as: Section 4.1, describing the datasets used in this experiment. Section 4.2 and 4.3, describing the traditional classification algorithm and features representations used to create classification models. The experiment begins with the baseline in Sections 4.4, and is followed by data resampling in sections 4.5, 4.6, and dimensionality reduction techniques in Section 4.7. In Section 4.8, we present a multi-class classification experiment followed by a discussion in Section 4.9. Finally, the conclusion in Section 4.10.

The experiments in this chapter were published in:

- Queiroz, A. L., Mckeever, S., and Keegan, B. (2019). Eavesdropping hackers: Detecting software vulnerability communication on social media using text mining. In *The Fourth International Conference on Cyber-Technologies and Cyber-Systems*, pages 41–48.

## 4.1 Datasets

Existing works on text mining within the security domain have evaluated classification models with a smaller number of datasets. Such an approach depends on the compatibility of models and the specific source they are applied to. In other words, the best model configuration created with Twitter data might not be suitable for Deep Web hacker forums and vice-versa. Consequently, our approach is focused on analysing the best configuration considering more than one dataset. This way, we are not evaluating the model based on a specific source, instead, our analysis is focused on an optimal configuration that can be applied to different sources.

The datasets used are D1 (hacker forum), D2 (Twitter) and D3 (marketplace). The specific purpose of each dataset and description can be seen in Section 3.2. Also, they are labelled using Partial Label Agreement (PLA) as described in Section 3.3.3, which considers the majority of votes to assign the definitive label to each message. The content of the messages in these datasets share the following common characteristics:

1. All posts are in some way related to computer technologies, including the security aspects of software, systems, network and protocols
2. All datasets have an imbalanced number of instances within classes, with fewer representing the positive (target class) than negative (non-target class)

In order to create the models using the mentioned dataset and labels, we considered two classification approaches: binary and multi-class. The binary classification is the most commonly used approach, in which a model classifies instances in two classes (positive or negative). However, for performing the binary approach with our dataset, we have transformed the "undecided" label into "yes" (positive) as we believe that these messages might represent a likely risk in a real-life situation and should be captured for further analysis. Afterwards, we compared the results of the binary experiment with the multi-class experiment to investigate which approach is more suitable for detecting software-vulnerability communication.

On the other hand, in multi-class classification, the model learns to classify more than 2 classes, which allowed us to use the label "no", "yes" and "undecided", separately. Additionally, we have decided to use One-vs-Rest (OvR) strategy for training the models as it provides similar accuracy compared to a commonly used strategy called One-vs-One (OvO), although its run-time is significantly less (Chmielnicki and Stąpor, 2016).

Finally, the description of the datasets for the binary classification task is seen in Table 4.1 and the experimentation is presented in Section 4.4. Also, the description of the datasets for the multi-class task is described in Table 4.2 and the experiment in Section 4.8, respectively.

Table 4.1: Dataset used in Binary Classification

| *ID* | *Source* | *No instances* | *Distrib. (pos/neg)* | *Avg. No. words* |
|---|---|---|---|---|
| **D1** | Hacker Forum | 1,682 | 10/90% | 50 |
| **D2** | Twitter | 1,921 | 15/85% | 13 |
| **D3** | Dark Web | 1,927 | 16/84% | 169 |

Table 4.2: Dataset used in Multi-class Classification

| *ID* | *Source* | *No instances* | *Distrib. (pos/neg/und)* | *Avg. No. words* |
|---|---|---|---|---|
| **D1** | Hacker Forum | 1,682 | 7/90/3% | 50 |
| **D2** | Twitter | 1,921 | 13/85/2% | 13 |
| **D3** | Dark Web | 1,927 | 11/85/4% | 169 |

**Data Pre-processing**

Data pre-processing is a commonly used step of text classification. These are necessary steps for reducing the noise and improving the accuracy of classifiers (Jianqiang and Xiaolin, 2017). However, the appropriate pre-processing methods are domain and language dependent (Uysal and Gunal, 2014). For this reason, we are using methods that have been widely used for text classification in English language, such as following:

- Change of message letters to lowercase
- Replace links such as https://examplelink.com to a mark [URL]
- We have replaced mention to users such as @users to a mark [USERNAME]
- For Twitter dataset, the RT (re-tweets) marks were removed

- Removal of stop-words (such as prepositions pronouns and common words). we have not performed this operation on Twitter datasets, as they are generally short conversations.

## 4.2 Traditional Classification Algorithm

As seen in Chapter 2, a considerable part of works on text classification has been using traditional classification algorithms. Among all options, we selected Support Vector Machine (SVM) and Naïve Bayes (NB). These algorithm are among the top choices in several different text classification tasks, e.g., sentiment analysis, new filtering, opinion mining, spam filtering, due to good results in prediction accuracy. The following subsections provide some characteristics of these algorithms in the context of our work.

### 4.2.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for classification tasks (Cortes and Vapnik, 1995). This algorithm is based on the maximal margin principle and known for achieving favourable performance with high dimensional data and text classification (Joachims, 1998, Suykens and Vandewalle, 1999).

SVM has been used as the first choice for solving several tasks in text classification domain, such as spam detection (Lee and Kang, 2019), sentiment analysis (Liu, 2017), online hate speech detection (Chen et al., 2017a). Also, this algorithm has been used for several security domain tasks, such as identification of malware's attack vector (Benjamin and Chen, 2013), mobile authentication through patterns of screen touch (Saravanan et al., 2014) and identification of malicious executables (Kolter and Maloof, 2006).

In this experiment, we apply the algorithm for detecting software vulnerability communications in social media and hacker forums. Moreover, we are using the SVM with linear kernel and the default value of C hyperparameters, given by the library (Sci-kit 0.2.20v). We choose to use the defaults to fairly compare the performance of the models throughout using all 5 different datasets.

### 4.2.2 Naïve Bayes (NB)

Naïve Bayes (NB) algorithm, is based on the Bayes' theorem, which is considered one of the most commonly used, simplest and efficient algorithms for text classification tasks (Lewis, 1998, McCallum and Nigam, 1998, Sebastiani, 2002). NB has been proved successful in a variety of applications, particularly for the field of text classification (Peng and Schuurmans, 2003, Sang-Bum Kim et al., 2006).

According to Bayes's theorem, Equation 1, NB is calculated as a posterior probability $p(C_k|x)$ of a class $C_K$ (with k = 1, ..., k) and an unseen example $x$ (composed by a set of features $x1, x2, x3, ..., xn$). In practice, the denominator, $p(x)$, is effectively constant for every data point in the training set. Additionally, the prename "Naïve" comes from the assumption that all features are calculated independently of each other (mutual independence property).

$$p(C_k|x) = \frac{p(x|C_k)(C_k)}{p(x)} \tag{1}$$

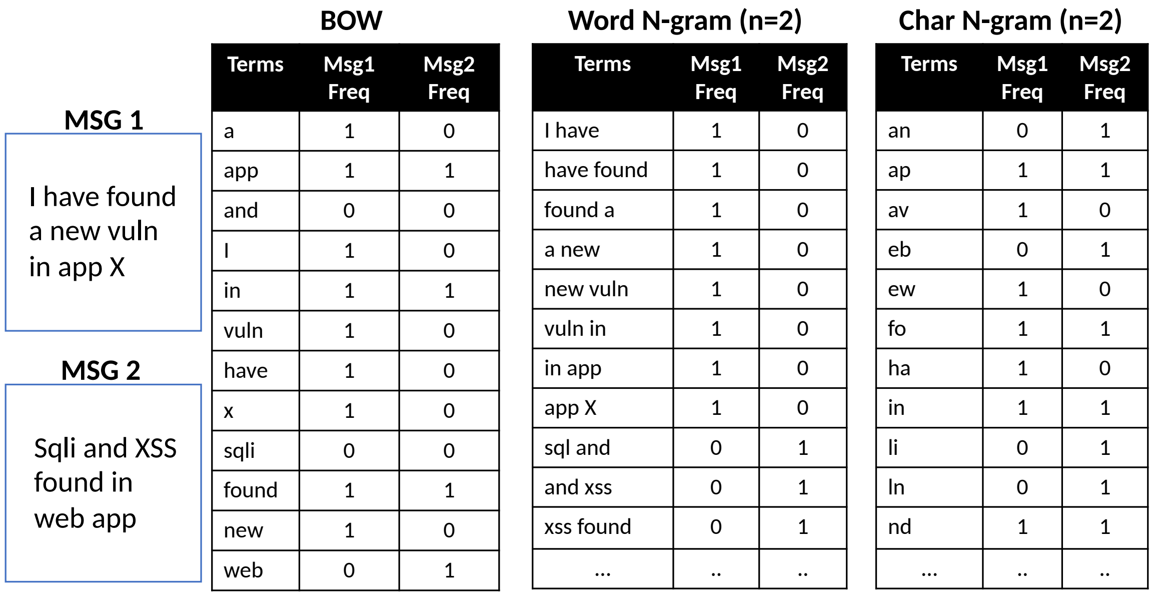
In classification models, this algorithm comes with a decision rule which the intuition resumes in "assign the unseen example with the most probable class" given by Equation 2.

$$\hat{y} = \underset{k\in\{1,\dots,k\}}{\operatorname{argmax}}\, p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{2}$$

## 4.3 Traditional Feature Representation

As previously mentioned in Section 3.4.2, we chose to use word features, which are generic to all datasets. The features representation used in this experiment are bag-of-words (BoW), words n-grams (W_ngram) and *char n-grams* (C_ngram).

In these representations, the words (or characters) of the messages of datasets are split into a defined set of tokens. Each message is now represented by a single vector containing a normalised occurrence (frequency) of these tokens. For the BoW approach, the text is tokenised into an unordered set of words, where each separate word represents a single feature. Whereas, for the *words n-grams* approach, which is considered an improvement upon BoW, we defined the size of the token by changing the value of *n*, which consists of N words occurring in sequence. Not unlike the previous representation, with *char n-grams*, the process is the same, however, more granular and it is based on characters of the words. Examples of this representation are seen in Figure 4.1.



**MSG 1**

I have found a new vuln in app X

**MSG 2**

Sqli and XSS found in web app

**BOW**

| Terms | Msg1 Freq | Msg2 Freq |
|---|---|---|
| a | 1 | 0 |
| app | 1 | 1 |
| and | 0 | 0 |
| I | 1 | 0 |
| in | 1 | 1 |
| vuln | 1 | 0 |
| have | 1 | 0 |
| x | 1 | 0 |
| sqli | 0 | 0 |
| found | 1 | 1 |
| new | 1 | 0 |
| web | 0 | 1 |

**Word N-gram (n=2)**

| Terms | Msg1 Freq | Msg2 Freq |
|---|---|---|
| I have | 1 | 0 |
| have found | 1 | 0 |
| found a | 1 | 0 |
| a new | 1 | 0 |
| new vuln | 1 | 0 |
| vuln in | 1 | 0 |
| in app | 1 | 0 |
| app X | 1 | 0 |
| sql and | 0 | 1 |
| and xss | 0 | 1 |
| xss found | 0 | 1 |
| ... | .. | .. |

**Char N-gram (n=2)**

| Terms | Msg1 Freq | Msg2 Freq |
|---|---|---|
| an | 0 | 1 |
| ap | 1 | 1 |
| av | 1 | 0 |
| eb | 0 | 1 |
| ew | 1 | 0 |
| fo | 1 | 1 |
| ha | 1 | 0 |
| in | 1 | 1 |
| li | 0 | 1 |
| ln | 0 | 1 |
| nd | 1 | 1 |
| ... | .. | .. |

Figure 4.1: Example of Traditional Features Representation (*BoW, Word and Char n-grams*)

Additionally, for these representations, we are using a range of 1 to 4 words/char tokens, N=(1,4), for W_ngram and C_ngram representations with the frequency of the given n-gram as weights for the feature vectors. By using a long range of n-grams (N=(1,4)), we are able to capture different contexts in the same message, since n=1 represents one single word, and n=2,3,4 represents a set of sequential words.

Moreover, very large and very small frequencies of words might occur throughout the posts of dataset. These large ranges might interfere with the decision boundary of the models. To avoid this interference, we have applied the *Min-Max* normalisation on these frequencies in range of 0 to 1, according to Equation 3. This method is commonly used in data mining research (Al Shalabi and Shaaban, 2006). The $v$ is the value before, and $v'$ is the value after normalisation. The $min_v$ and $max_v$ are the minimum and maximum values of the distribution, whereas $min_{new}$ and $max_{new}$ are the values used to scale the features, in our case, 0 and 1, respectively.

$$v' = \frac{v - min_v}{max_v - min_v}(max_{new} - min_{new}) + min_{new} \tag{3}$$

## 4.4 Baseline Model Configuration

In order to define the best (baseline) configuration for the models, we have created them by permutation of two classification algorithms and three features representations. Further, these models are evaluated in 3 different datasets (D1, D2, D3).

The algorithms of choice are SVM and NB, and the features representations are *BoW, Words n-gram and Chars n-gram*. The values of n for the *word/char n-grams* are in the range of 1 to 4, which means that n-gram of 1, 2, 3 and 4 are part of the features set.

The values recorded for Avg. Class Accuracy and Positive Recall of the model for each individual dataset is presented in Table 4.3. It is seen that the best feature representation is *char n-gram* for SVM algorithm, in both metrics, whereas for Naïve Bayes, BoW has shown better performance in 2 of 3 datasets, D2 and D3. Additionally, to investigate the best configuration of models, we have averaged the Avg. Class Accuracy of models throughout all datasets. These results can be seen in Figure 4.2 as a modified boxplot graph, where the middle line of the box represents the mean of Avg. Class Accuracy metric (instead of median) with the results recorded by each dataset plotted with a different mark in the graph.

Table 4.3: Average Class Accuracy and Positive Recall of SVM and Naïve Bayes Models per Dataset

| | | **SVM** | | **Naïve Bayes** | |
|---|---|---|---|---|---|
| *Dataset* | *Feat. Rep* | *Avg. Class Acc* | *Recall (positive)* | *Avg. Class Acc* | *Recall (positive)* |
| **D1** | BoW | 0.67 | 0.38 | 0.57 | 0.15 |
| | W_ngram | 0.65 | 0.32 | **0.59** | **0.20** |
| | C_ngram | **0.71** | **0.46** | 0.55 | 0.11 |
| **D2** | BoW | 0.85 | 0.71 | **0.84** | **0.69** |
| | W_ngram | 0.76 | 0.53 | 0.80 | 0.61 |
| | C_ngram | **0.86** | **0.76** | 0.83 | 0.68 |
| **D3** | BoW | 0.73 | 0.52 | **0.75** | **0.56** |
| | W_ngram | 0.71 | 0.46 | 0.63 | 0.28 |
| | C_ngram | **0.76** | **0.60** | 0.73 | 0.56 |

In terms of averaged result per model (mean line in Figure 4.2), we see that SVM outperforms Naïve Bayes algorithm in 2 of the 3 presented models, the best combination using Naïve Bayes is BOW (NB+BOW) with an average 0.72 of Avg. Class Accuracy. The best performance is achieved with SVM+C_NGRAM and SVM+BOW, with 0.78 and 0.76 of Avg. Class Accuracy respectively. For these two best models, we have the Friedman Statistical test to determine whether there is any difference on the recorded results. With p-value=.097 (for alpha = .05), we fail to reject the null hypothesis, meaning that there is no statistical certainty that any model is outperforming any another. Considering this, the baseline preferred for the remainder of this work is the SVM+BOW as this model is less expensive in terms of computational resources,

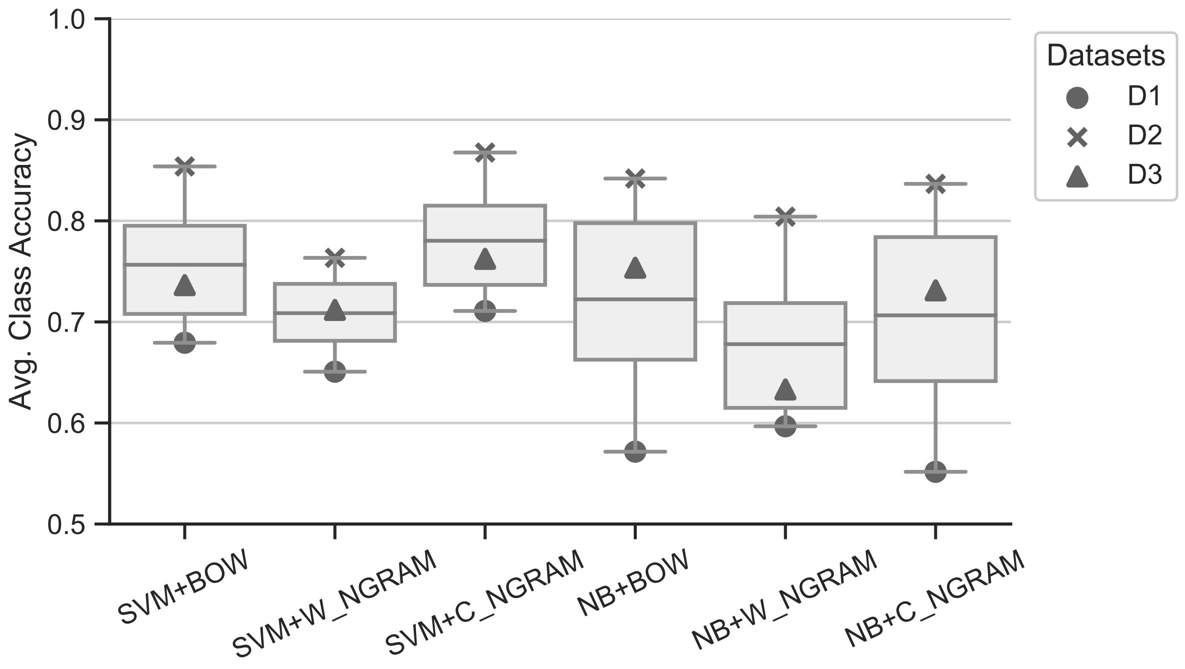


Figure 4.2: Baseline results.

which consumes less processing power and memory, thus is trained faster than SVM+ C_NGRAM.

## 4.5 Imbalanced Datasets

As seen in Table 4.1, the datasets used for the binary experiment in this research have an imbalanced number of instances, with the majority class being from the negative class (non-target), which also represents at least 5 times more instances than positive instances. According to Santos et al. (2018), a model trained with imbalanced data has shown lowest performance compared to a model trained with balanced data. Without sufficient knowledge to learn from the minority classes (positive), classifiers may over assign instances to the majority classes (negative)(Ganganwar, 2012).

In this research, one of the main metrics used to evaluate the model is Positive Recall, as mentioned in Section 3.4.3. However, these models present a low Positive Recall compared to Negative Recall in all datasets, as seen in Figure 4.3. In order to increase the Positive Recall, we use random over-sampling to increase the number of positive instances on datasets D1, D2, D3. This technique has been seen as a viable alternative to enhance the positive recall of models trained on imbalanced datasets (Chen et al., 2017a). In our experimental procedure (Section 4.6), we randomly resampled each fold three times, recording the average of the results for each run to minimise any random selection influence. It is worth highlighting that we have not applied this technique to the test fold data (evaluation set), as this fold should represent the way data would appear in a real-life application of the model.

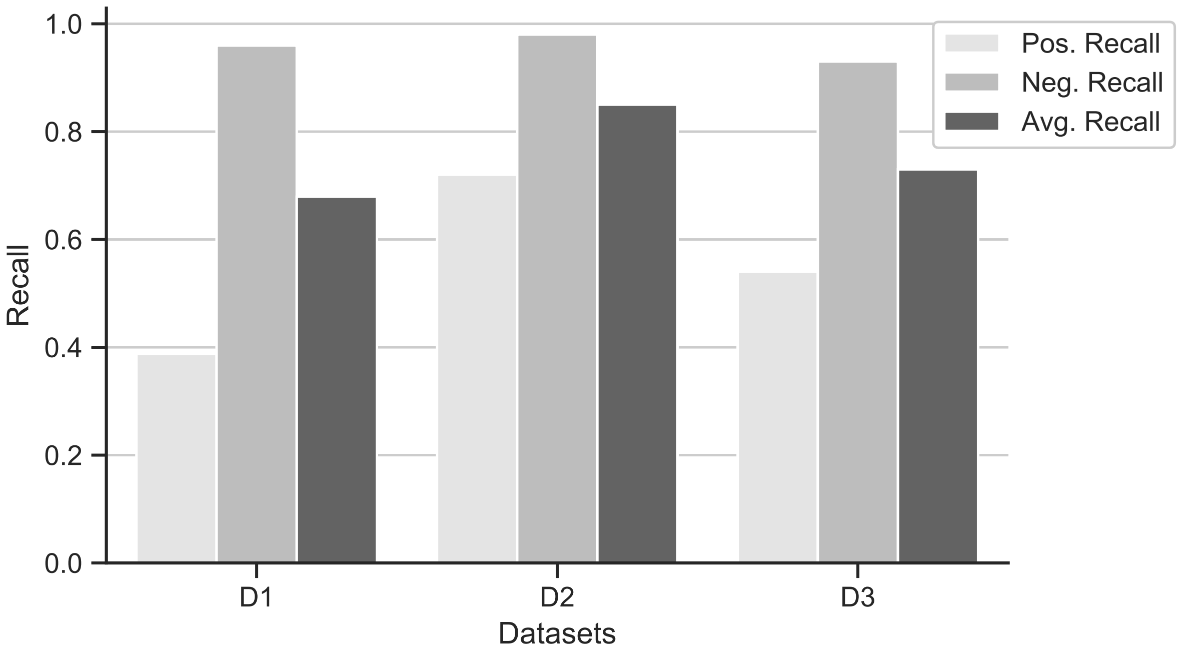


Figure 4.3: Pos., Neg., and Avg. Recall for SVM+BOW

## 4.6 Dataset Resampling

Chen et al. (2017a) has demonstrated that oversampling techniques can increase the Positive Recall. However, an excess of oversampling can lead to an overfitting of the model (Santos et al., 2018). For finding the optimal oversampling size, we have explored different resampling proportions for the positive class by using the *first optimal rule* (Queiroz et al., 2019).

This rule consists of setting the optimal point as being the one that most improves

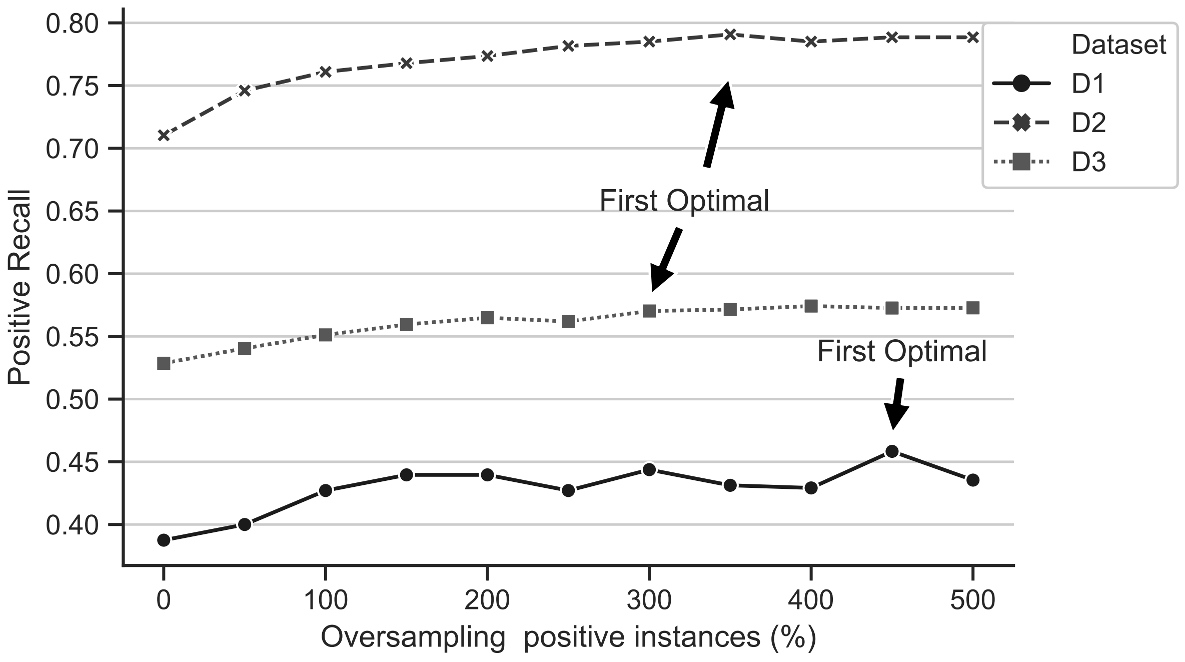


Figure 4.4: Optimal Point

the positive recall and records a minor proportion of resampled instances. Following this rule, D1, D2 and D3 have their first optimal point set as 450%, 350% and 300% respectively as seen in Figure 4.4.

With this technique, we achieve a 6% increase in Positive Recall for D1 and D2, and 3% for D3, compared to the baseline model (without re-sampling), which represents an average increase of 5%. Finally, the new proportion of the classes is shown in Table 4.4.

Table 4.4: Re-Sampled Dataset

| | *D1* (+/-) % | *D2* (+/-) % | *D3* (+/-) % |
|---|---|---|---|
| **Before** | (10/90) | (15/85) % | (16/84) % |
| **After** | (44/56) | (32/62) % | (37/63) % |

## 4.7 Dimensionality Reduction

Traditional classification approaches using textual data usually contain a large number of features (high-dimensional). However, not all features have equal importance for improving the performance of a model. On the contrary, a large number of useless features prevent the model from properly learning the patterns of the data on small datasets, which is a phenomenon known as "Curse of Dimensionality" (Bellman et al., 1957).

The use of Dimensionality Reduction (DR) is often applied to solve this problem. These techniques are important for maintaining the computational efficiency of models, size compression and it is also helpful in other tasks such as visualisation (van der Maaten et al., 2008).

Thus, the aim of this section is to apply DR techniques for removing such features that do not contribute to the overall accuracy of the model. We have used three different techniques to perform this reduction. The experiment was performed in a structured way such that we remove the features to a minimum while maintaining the best accuracy of the model. The experiment starts with a basic technique called Document Frequency (DF) reduction, and, on top of that, we apply Chi-square as Feature Selection technique and Singular Value Decomposition (SVD) as Feature Extraction.

### 4.7.1 Document Frequency

As mentioned, the models in this experiment are high-dimensional and are represented with a sparse vector (larger number of zeros) in all trained datasets. In Table 4.5, we see the number of features for each model by feature representation.

For many learning algorithms, including the one described in Section 4.2.1, training and classification time increases directly with the number of features. In addition, higher numbers of features (high-dimensionality) combined with fewer instances

Table 4.5: No. of Feature Per Feature Representation

| Text Representation | *D1* | *D2* | *D3* |
|---|---|---|---|
| **BoW** | 9,422 | 4,880 | 18,119 |
| **Word n-grams** | 168,082 | 49,610 | 542,259 |
| **Char n-grams** | 96,063 | 36,372 | 104,981 |

might incur in a phenomenon called "Curse of Dimensionality" (Bellman et al., 1957), which impacts negatively on classifier accuracy. A simple technique to reduce the number of features is the use of document frequency (DF) reduction (Forman, 2003). DF reduction uses the number of words that occur within the documents (messages in social media) and removes it, so it is no longer a feature of the model, by defining a threshold of most and least commonly recurring words. By removing the most common words, we remove words which offer less meaning to the sentence, so called stop words (nouns, articles, pronouns), whereas by removing the least common words, we remove rarely used words.

In Figure 4.5, we demonstrate that we can achieve a reduction of the number of features to at least 50% (0.5) for all datasets by adjusting threshold to 0.1, which represents the exclusion of 0.1% top and bottom frequent words. As a result, we excluded the most frequent words, appearing in $>$ (more than) 20%, and least frequent words, appearing in $<$ (less than) 0.1% of the messages.

The performance of the model before and after reduction can be seen in Figure 4.6, where we provide a comparison of the accuracy of the new model with the optimal values achieved so far (model without DF reduction). It is seen that D1 and D2 recorded the same optimal result compared to the previous model, while D3 has recorded an increase of 1%. Thus, we are able to reduce the dimensionality of the model, while maintaining the detection performance.

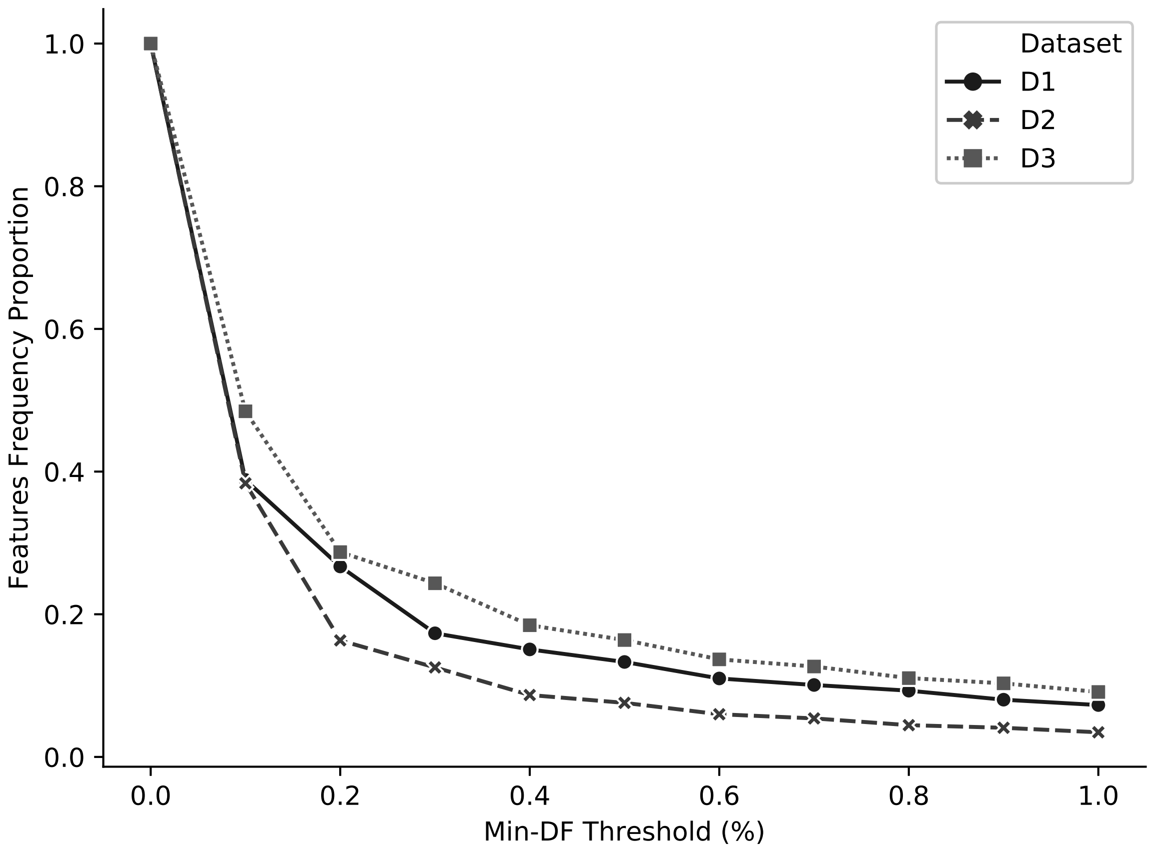


Figure 4.5: Feature Reduction

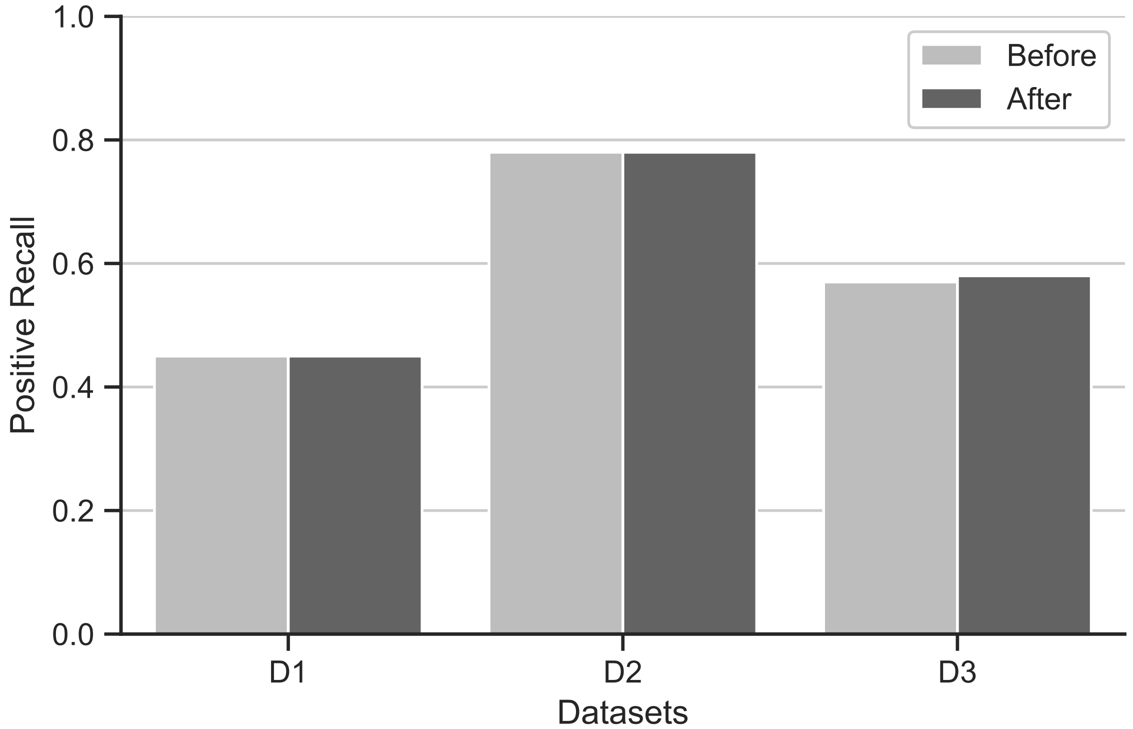


Figure 4.6: Comparing Positive Recall of Models (Before and After DF Reduction)

### 4.7.2 Feature Selection and Feature Extraction

To further reduce the dimensionality of the model, we have investigated two other approaches, namely, Feature Selection and Features Extraction. We have applied them on top of the previous DF reduction. In the following, we briefly explain each approach and its methods:

- **Feature Selection** involves a ranking for determining the best subset of the existing features. Typically, it uses algorithms that correlate each subset of feature with the target class label. Thus the selection of the best subset is based on the top-ranked ones. Furthermore, it helps eliminate the noise of less predictive features and significantly reduces the dimensionality without losing classification performance. In this part of the experiment, we are using the Chi-Square technique on top of the previous reduced configuration (DF reduction). This technique has been successfully applied to reduce dimension in other approaches that use SVM algorithm (Bahassine et al., 2020, Chen et al., 2017a, Ikram and Cherukuri, 2017).
- **Feature Extraction** involves a transformation of the existing feature to a set of alternative, more compact, features. This provides retention of as much information as possible. A Commonly used method includes the Singular Value Decomposition (SVD). SVD transforms the data into a reduced feature space and as a characteristic, it captures most of the variance in the data. This method has been used in other text mining problems, which have been providing promising results (Chen et al., 2017a, Harrag and El-Qawasmah, 2009, Kadhim et al., 2014)

Through experimentation, we reached the minimum reduction using Chi-square of 50% of the features, whereas for SVD, we were able to reduce to 10% of its current size, in other words, a reduction of 90%. It is worth highlighting that, Chi-Square excludes 50% of the features, whereas SVD compacts all features in 10% of the feature space. In Figure 4.7, the results indicate that we can use both techniques to further reduce the dimensionality of the model without decreasing performance. However, for DF + SVD, there is a minor reduction in Positive Recall (about 1%).

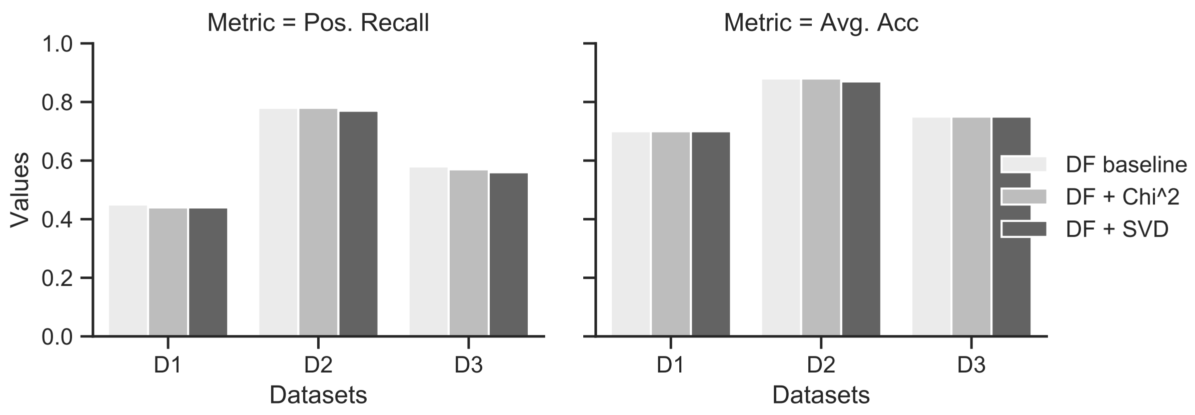


Figure 4.7: DF, Chi-square and SVD Dimensionality Reduction with SVM+BOW

## 4.8 Exploring as a Multi-class Problem

Up to now, we have analysed the performance of models using a binary classification approach, where we decided to use the dataset labels *Yes* and *Undecided* as Positive class and *No* as Negative class. Additionally, we have analysed oversampling technique for increasing the accuracy of the model. Furthermore, we applied feature selection and features extraction to reduce the dimensionality, thus providing a more efficient model.

In this section, instead of binary, we perform an experiment using multi-class classification approach, where the model is trained to predict more than two classes. In our case, we are predicting three classes separately, *Yes*, *No* and *Undecided*. This experiment is performed for investigating whether multi-class approach yields better performance than binary approach for our classification task.

Similar to the previous experiment, we have observed that the classes also present an imbalanced number of instances with minor messages under *Undecided* and *Yes* classes (Table 4.2). In order to deal with the imbalanced nature of the instances, we have resampled the minority instances using the *first optimal rule* previously mentioned in

Section 4.6. We have also applied the same dimensionality reduction as in binary classification (seen in Section 4.7).

Therefore, as it is not trivial to compare multi-class with the binary classification approach, we have grouped the *Yes* and *Undecided* labels together in a new class called **Risk Threat**. This new class contains messages that represents true malicious communication (*Yes*) and those that are not completely discarded from being malicious (*Undecided*). We acknowledge that, with a security perspective, *undecided* messages carry a likely risk, thus they need further inspection, for this reason we treat them as belonging to the Risk Threat class.

In order to compute the scores of Risk Threat class, we consider the correct classified instances as: the *Yes* and *Undecided* instances that are predicted as yes or undecided. The calculation of this score is given by equation 4, where we sum the correctly classified instances divide by the total of true labels of Yes and Undecided classes.

$$RiskThreatRecall = \frac{(TP_{yes} + TP_{undecided})_{yes} + (TP_{yes} + TP_{undecided})_{und}}{TotalTrue_{yes} + TotalTrue_{und}} \tag{4}$$

Table 4.6: Multi-class Classification Confusion Matrix Results

D1

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | No | Yes | Und | Total |
| True Label | No | 147 | 3 | 2 | 152 |
| | Yes | 5 | 4 | 1 | 10 |
| | Und | 3 | 1 | 1 | 5 |

D2

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | No | Yes | Und | Total |
| True Label | No | 162 | 1 | 0 | 163 |
| | Yes | 3 | 20 | 0 | 23 |
| | Und | 3 | 2 | 0 | 5 |

D3

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | No | Yes | Und | Total |
| True Label | No | 156 | 6 | 2 | 164 |
| | Yes | 8 | 11 | 1 | 20 |
| | Und | 4 | 1 | 1 | 6 |

Table 4.7: Recall For Multi-class and Binary Approaches

| *ID* | *Source* | *Recall (Multi-class)* | *Recall (binary)* |
|---|---|---|---|
| **D1** | Hacker Forum | 0.46 | 0.44 |
| **D2** | Twitter | 0.78 | 0.77 |
| **D3** | Dark Web | 0.54 | 0.56 |
| | Avg | 0.59 | 0.59 |

In Table 4.6, we present the confusion matrix per dataset. This table shows the prediction made by the model as well as the true label of each class. Using these values, we provide the Recall for the Risk Threat class in Table 4.7. Additionally, a comparison of the previous binary with the multi-class experiment is presented.

It is seen that, on average, both experiments lead to the same result, 0.59 recall. The results in this section strongly suggest that using *Undecided* classes separately from *Yes* does not enhance the performance of the model in general. However, in terms of individual source, we have found that the model created over D1 (Hacker Forum) and D2 (Twitter) source, has better recall using the multi-class approach. Comparing these dataset with D3 (Dark web), we noticed that D1 and D2 contain, on average, fewer words per message than D3, and also have less proportion of undecided classes, which D3 has higher proportion, 4%, compared to D1, 3%, and D2, 2%. We believe that multi-class should be applied to datasets with fewer numbers of "undecided" instances and have at maximum an average of 50 words per message.

## 4.9 Discussion

Considering the results per dataset (Table 4.8), we see that the best result is achieved by the model trained in dataset D2 (re-sampled), with 78% of recall, while D1 and D3 (re-sampled) have recorded very poor performance, 45% and 57% of recall, respectively. D2 represents the dataset trained in Twitter Security Experts posts, where the messages are aimed at a broad audience, whereas D1 and D3 are focused on expert users, in which technical terms are more frequently used. We believe that the use of technical terms influence the performance of the model. Therefore, the combination of algorithm and features representation that provided this result (SVM+BOW) is suitable for datasources where the use of technical terms are limited, such as D2 (Twitter). However, it is not suitable for a more technical-focused sources, such as D1 (hacker forums) and D3 (marketplace).

This situation is an opportunity to further investigate other text mining techniques that can improve on the prediction of the target class messages in all types of sources. In the next chapter, we provide an analysis of the use of Word and Sentence Level Distributed Representation as features representation. Additionally, we have investigated the use of Neural Network-based algorithms as classifiers, for instance, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

Table 4.8: Summary of Results - Min-Max [0,1]

| | Metric | Baseline | Re-sampled | DF | DF+Chi2 | DF+SVD |
|---|---|---|---|---|---|---|
| **D1** | Avg. acc | 0.68 | 0.70 | 0.70 | 0.70 | 0.70 |
| | Pos. recall | 0.39 | 0.45 | 0.45 | 0.45 | 0.44 |
| **D2** | Avg. acc | 0.85 | 0.88 | 0.88 | 0.88 | 0.87 |
| | Pos. recall | 0.72 | 0.78 | 0.78 | 0.78 | 0.77 |
| **D3** | Avg. acc | 0.73 | 0.75 | 0.75 | 0.75 | 0.75 |
| | Pos. recall | 0.54 | 0.57 | 0.58 | 0.57 | 0.56 |

## 4.10 Summary and Conclusion

The principal goal of this chapter is to analyse the traditional text mining techniques to detect software-vulnerability-related communication. Whereas, the sub-goal is to investigate the impact of features representations, balancing and feature reduction techniques for the performance of classification models. We performed a systematic experiment using different combinations of algorithms, features representations and dimensionality reduction to create efficient and accurate models.

The algorithms used were the traditional SVM and Naïve Bayes, as they are suitable for a large number of word features and commonly used in text-classification tasks. For feature representation, we used the traditional *bag-of-words, word n-grams and char n-grams*. Also, in dimensionality reduction, we have used document frequency reduction, feature selection (chi-square) and feature extraction (SVD) techniques for improving the efficiency of the models.

These experiments were performed in two different classification approaches: binary and multi-class classification. In the former approach, the labels "yes" and "undecided" were assigned as positive instances, whereas the latter approach treated them separately. The results have shown that, in general, both approaches are equally suitable for the task of detection of software vulnerabilities communication.

Furthermore, it is accepted that, due to the few instances of positive class (malicious) compared to negative class (non-malicious), the model over-assigns the negative instances, which leads to poor performance in terms of Recall (Santos et al., 2018). For this reason, we investigated the use of augmentation techniques (random oversampling) to increase the number of instances of the minority class and thus improve the recall metric of the model.

All evaluations were performed through three different datasets, each one collected from a different internet domain (Surface Web, Deep Web, and Dark Web). The summary of the achievements for all steps taken in this work can be seen in Table 4.8. With respect to them, it has been concluded:

- On average, the model created with SVM performs better than the Naïve Bayes algorithm. SVM using *char n-grams* and BoW features representations are the best combination of the classification models tested in this chapter.
- The Binary classification approach (which uses undecided as positive instances) and multi-class (which used all the 3 classes separately), on average, leads to the same performance. Only D1 and D2, which is a dataset with a lower proportion of undecided compared to D3, has benefited from multi-class strategy compared

to the binary classification. Also, D1 and D2 datasets contain a lower average of word per message. We believe that these two mentioned characteristics are key for using a multi-class approach for detecting hacker communication.

- The random sampling technique has shown to be useful for training models with imbalanced quantity of instances within the classes. In this experiment, we have an increase in Positive Recall of 5%, on average, by oversampling the minority class. Models trained in D2 and D3, achieve the best positive recall (with less oversampling) by increasing the number of minority class from 350% and 300%, respectively, whilst D1 has reached its best in 450% resampling.
- The same detection performance of the models can be achieved by reducing the number of features. With DF reduction, by removing the least and the most often features, we achieved a drop of 50% of the total number of word features, while maintaining the same classification performance. To further reduce the dimensionality of the model, Chi-Square and SVD techniques can be used on top of DF at the levels of 50% and 10% respectively.

CHAPTER 5

# Detection of Software Vulnerability-related Communication Using Deep Learning-based Models

In Chapter 4, we see that traditional algorithms assume that data can be divided by a linear decision boundary. However, for some text classification tasks, these linear models do not achieve their best performance (Kamath et al., 2018). Consequently, the use of Deep Learning (DL) algorithms has been presented as an option for improving the accuracy of text classification, as they are more suited for non-linear and complex data (Abiodun et al., 2018). As a result, in this chapter, we investigate how DL algorithms can be combined with Language Models (LM) for improving the detection of software vulnerability communication. Throughout the remainder of this chapter, we use the terms Word Embedding (WEMB) and Sentence Embedding (SEMB) to refer to two distinct categories of LM.

Recently, there has been a surge of state-of-the-art LM in NLP tasks. They are commonly categorised as Word/Sentence Distributed representation. Additionally, to improve the performance of models in text classification tasks, they can be combined with different DL architectures, including the well-known Recurrent Neural Networks (RNN) and text-based Convolutional Neural Network (CNN) (Kim, 2014).

Moreover, as mentioned in the literature review, Chapter 2, the WEMB and SEMB Language Models are considered an evolution over traditional representation, such as *bag-of-words* (BoW) and *Word- or Char- n-grams*. A well-known characteristic of such models is the ability to infer the semantic information of word (or sentence) based on capturing contextual word usage as part of the embedding training task. As a result, they have shown better detection performance in a range of different downstream text classification tasks, such as spam detection (Lee and Kang, 2019), abusive content detection (Chen et al., 2017a) and news categorisation (Wu et al., 2017).

To determine the optimum classification model for out detection task, we have combined and evaluated different algorithms (i.e., CNN, RNN and SVM) with a variety of LM (i.e., *Word2vec, Glove, Sen2vec, Inferset, SentEncoder*), using five different datasets collected from different sources (i.e., Forums, marketplace and Twitter). For WEMB and SEMB, we have used pre-trained models trained in large corpus extracted from internet content such as Wikipedia, Twitter, and News. Moreover, we have compared the pre-trained WEMB with our trained WEMB, which uses a large corpus of software vulnerability descriptions. This content was extracted from NIST Vulnerability Database (NVD), a well-known database used by a security specialist to prioritise security updates in their software assets.

This chapter is organised as follows: In Section 5.1, we begin by describing the characteristics of the dataset and the steps taken before the creation of the model, for instance, pre-processing and oversampling. Afterwards, in Section 5.2, we explain the selected language models that will act as a feature representation for the classification model, and, in Section 5.3, we describe the CNN and RNN algorithms used to create the DL-based classifiers. In Section 5.4, we provide the baseline model, which is an extension of the previously experiment done in Chapter 4. We use these baseline models to compare the performance with the models presented in Section 5.5, which uses Word Embedding as feature representation. Furthermore, in Section 5.5.2, we train a language model in a software-security vocabulary context to compare with a general-purpose language model in a downstream classification task. Furthermore, we investigate the use of SEMB models with traditional SVM classifiers in Section 5.6, and in Section 5.7, we investigate the accuracy of Deep Learning-based algorithms as classifiers for detection of software vulnerability communication. Finally, in Section 5.8, we present a discussion and, in Section 5.9, the summary and conclusion of this chapter.

The experiment in this chapter was published in:

- Queiroz, A. L., Mckeever, S., and Keegan, B. (2019). Eavesdropping hackers: Detecting hacker threats: Performance of Word and Sentence Embedding models in identifying hacker communications. In *The 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science*, volume 2563, pages 116–127, Ireland.

## 5.1 Datasets

In this experiment, we used the 5 datasets (D1, D2, D3, D4 and D5) from different hacker community sources. Each message was labelled by the PLA scheme as described in Section 3.3.3 and the purpose, content and usage of each source are described in Section 3.2. The messages within these datasets share the following common characteristics:

1. All posts are in some way related to computer technologies, including the security aspects of software, systems, network and protocols.
2. All datasets have an imbalanced number of classes, containing fewer positive instances (target class) than negative instances (non-relevant target class)
3. The purpose of these messages is the sharing knowledge about hacking/hackable technologies among peers

In Table 5.1, we outline the description of the dataset used in the following sections (before resampling).

Table 5.1: Dataset Description (Before Resampling)

| *ID* | *Source* | *Type* | *No. of instances* | *Distrib. (pos/neg)* | *Avg. words per msg.* |
|---|---|---|---|---|---|
| **D1** | Hacker Forum | Deep web | 1,682 | 10/90% | 50 |
| **D2** | Twitter | Surface web | 1,927 | 15/85% | 13 |
| **D3** | Marketplace | Dark Web | 1,921 | 16/84% | 169 |
| **D4** | Hacker Forum | Deep web | 1,966 | 13/87% | 78 |
| **D5** | Hacker Forum | Deep web | 1,974 | 5/95% | 68 |

**Data Pre-processing**

The pre-processing procedures in this experiment are similar to what was done with traditional models, Chapter 4. All steps to clean and reduce the input noise have been already described in Section 4.1 (Data Pre-processing).

### 5.1.1 Random Oversampling the Positive Instances

As seen in Chapter 4, the random oversampling technique has provided an improvement of recall measure on traditional models trained with an imbalanced number of instances. As a result, we have applied the same method for all experiments in this chapter. As seen in Table 5.1, the number of instances in all 5 datasets is imbalanced, with the positive class under-represented relative to the negative class. Therefore, we have oversampled the positive instances by 350% in all datasets, which is the median optimal values found in Queiroz et al. (2019). After re-sampling, the new ratio of instances is shown in Table 5.2. The oversampling has provided a more balanced dataset.

Table 5.2: Dataset Description (After Resampling)

| | *D1* (+/-) % | *D2* (+/-) % | *D3* (+/-) % | *D4* (+/-) % | *D5* (+/-) % |
|---|---|---|---|---|---|
| **Before** | (10/90) | (15/85) | (16/84) | (13/85) | (05/95) |
| **After** | (34/66) | (45/55) | (47/53) | (41/59) | (19/81) |

## 5.2 Word and Sentence Level Distributed Representation

In this section, we provide the language models selected for composing our classification model. They are divided into two categories, WEMB and SEMB as seen in Chapter 2. We selected a portion of the language models found in the literature. In the following, we detail our choice for these models and its characteristics.

### 5.2.1 Word Embedding (WEMB) models:

The WEMB's principal purpose is the creation of an "embedding" vector in which similar words can be found near to each other in a vector space. Also, these models are commonly used as input of text classification algorithms in downstream task classification. WEMB models can be categorised according to the creating task, which is by *prediction* task or using words *co-occurrence* matrix. The former uses Neural Network architecture, whereas the latter perform a factorisation of the co-occurrence matrix. In Figure 5.1, we provide examples of models per category.

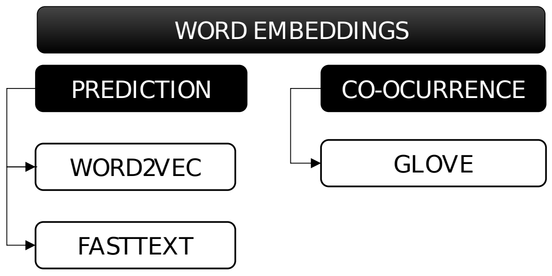


Figure 5.1: Word Embedding Categories

The most predominant models in the literature are *Word2vec* (Mikolov et al., 2013) and *Glove* (Pennington et al., 2014). Differently from the classical *bag-of-words* and *n-grams* language model, these models adjust vector values according to the word-context information. Moreover, *word2vec* calculates local-context information of words, while *Glove* provides two context-information, local and global. Furthermore, similarly to the traditional BoW feature representation, the drawback is that these models do not account for the order in which the word appears, which impacts on the distinction of sentences made up of the same words in a different order. Another well-known models is the *FastText* language model (Bojanowski et al., 2017). Similarly to *word2cec*, this model is created by an unsupervised prediction-based task and uses the same fundamental architecture of skip-gram, although it treats the words as a *bag of char n-gram* instead of unique words.

In this experiment, we have decided to use the most relevant model from each category, *prediction* and *co-occurrence* (as seen in Figure 5.1). This selection allows us to have a representative idea of how the models under these categories contribute to the classification performance in our specific task. As a result, we have selected the *word2vec* model, as it represents the first and most used word-similarity model under the prediction category. We have also selected *Glove*, which was the first model presented under the co-occurrence category. For representing the entire message (hacker communication), we are using a technique called *averaging* (Wieting et al., 2015), which is the simplest approach and has provided positive performance. This technique consists of averaging vectors according to the words occurring in the messages.

### 5.2.2 Sentence Embedding (SEMB) models:

The basic difference regarding SEMB to WEMB is that this language model is considering the entire sentence to create the embedding vector. This way, similar sentences are mapped to a close vector space. With this, the model overcomes one of the limitations of WEMB, the word order problem Le and Mikolov (2014). Thus, in SEMB, sentences with the same word in a different position (thus, different meaning) are represented by different vector values. Similarly to WEMB, SEMB can be categorised by the method they were created, that can be: *Unsupervised, Supervised and Multi-task. Multi-task represents the model created with Unsupervised and Supervised approaches*. Figure 5.2 presents these categories with examples of these models.

In this experiment, we have decided to select one of each category, which are: *Sent2Vec* (Le and Mikolov, 2014) (unsupervised), *InferSent* (Conneau et al., 2017) (supervised)

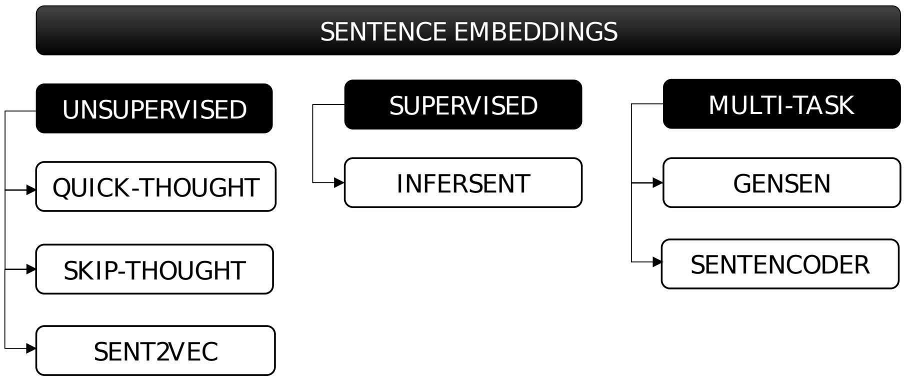


Figure 5.2: Sentence Embedding Categories

and *SentEncoder* (Cer et al., 2018) (multi-task).

## 5.3 Deep-learning Classification Algorithms

Besides the traditional SVM algorithm used in the baseline results and discussed in Section 4.2.1, we have also investigated two advanced Neural Network-based algorithms, the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), briefly introduced in Chapter 2 (Section 2.2.3). In this section, we explain the intuition behind the algorithms as well as the configuration used in our experiment.

### 5.3.1 Convolution Neural Network (CNN)

Convolution Neural Network has been providing remarkable results in several tasks such as image classification (Krizhevsky et al., 2017, Lecun et al., 1998) and speech recognition (Huang et al., 2014, Palaz et al., 2015). However, due to the successful results in image classification, this architecture was adapted for Natural Language tasks, as text classification (Amir et al., 2016, Gan et al., 2016, Ren et al., 2016, Young et al., 2017).

In this experiment, we adopted the text-based CNN architecture of Kim (2014) work and the hyper-parameters values are based on a study done by Zhang and Wallace (2015). In the latter work, the authors have provided an investigation of the optimal hyper-parameters by evaluating the performance of a CNN model across nine sentence classification tasks, for instance, sentiment analysis, customer review, opinion polarity and irony detection. The representation of this architecture is exemplified by Figure 5.3, where the input layer is represented by WEMB values of words in a sentence. In this example, we use a 7x5 matrix, where 7 is the number of words in a sentence and 5 the embedding values of the word (dimensions). The WEMB used is one of those seen in Section 5.2.1. Afterwards, these values are mapped (reduced) through a variety of convolutional filters (sizes 3, 4, and 5) to a small vector. In sequence, these values are reduced more using an operation called max-pooling. Finally, these max-pooled values are concatenated together and connected to a final *softmax* layer function, which gives the probability of the message being from a target or non-target class.

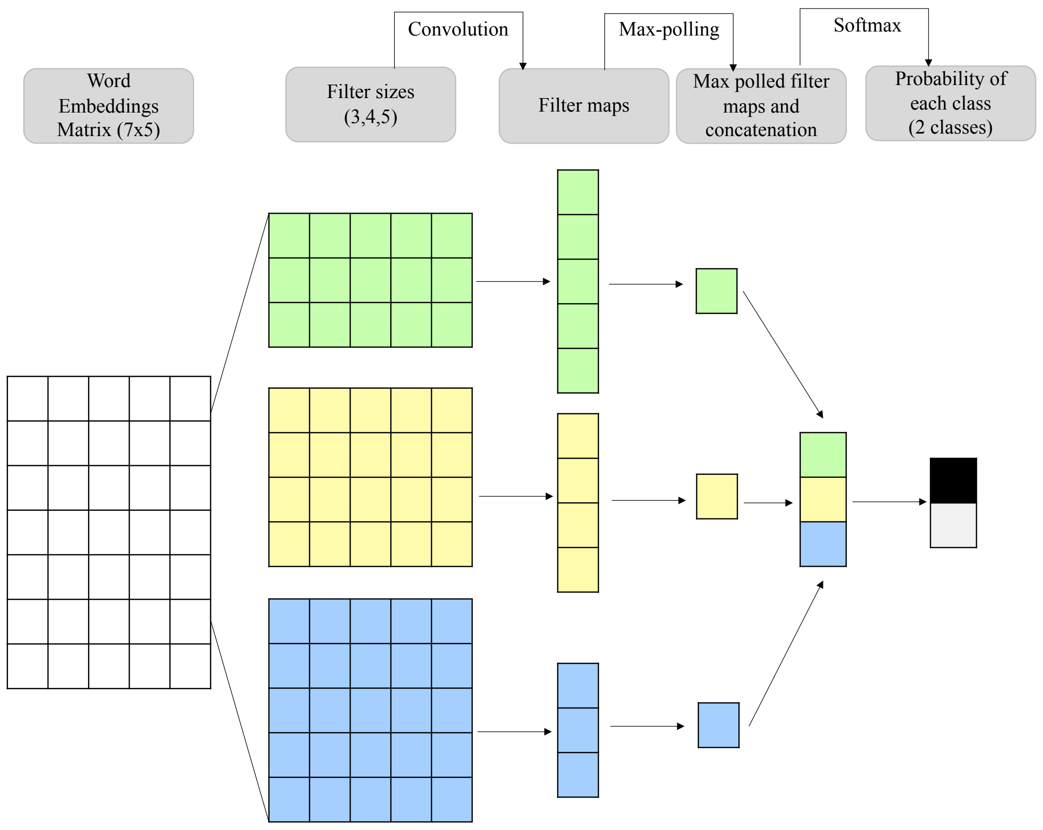

Convolution
Max-polling
Softmax
Word Embeddings Matrix (7x5)
Filter sizes (3,4,5)
Filter maps
Max polled filter maps and concatenation
Probability of each class (2 classes)

Figure 5.3: CNN Architecture (Based on Zhang and Wallace (2015))

CNNs were originally used for image classification. As a consequence, the input is usually a fixed-size image matrix. However, with text, we have a variable-size matrix, as a sentence might have different quantities of a word. To solve this problem, we have added a technique called zero-padding, applied to short posts to achieve the defined fixed-size.

As seen in Nam and Hung (2019), this technique might affect the efficiency of the model in terms of accuracy and training time. To define only the necessary padding to short sentences, we define the fixed-size input as the size of the longest sentence excluding the outliers (very long sentences). Figure 5.4 shows a comparison of results using fixed-length input either maintaining or excluding the outliers. It is seen that excluding the outliers provides better for permanence in the majority of the datasets. In CNN + *Glove* models, this procedure has helped to improve in 3 of 5 datasets (D2, D3 and D5), while for CNN + *word2vec*, it has helped to improve 2 of 5 datasets (D2 and D5). D3 and D4, however, are tied with the same performance.

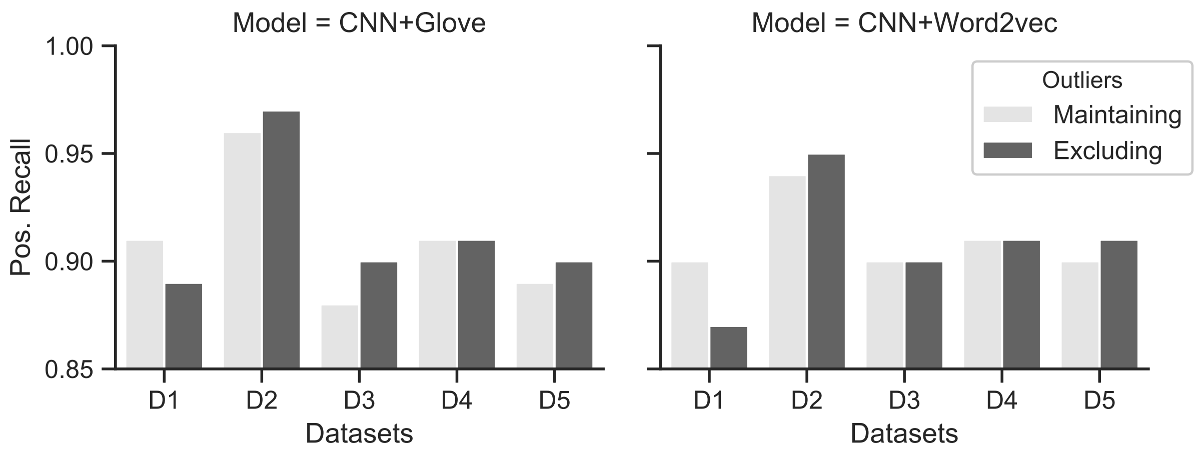


Figure 5.4: Comparison of Strategies for Finding the Fixed Input Length for CNN

Finally, we used the *Adam* optimiser instead of the popular *Stochastic Gradient Descent* (SGD), as it has presented more efficiency with faster convergence results (Kingma and Ba, 2014). As loss function, we have used the categorical cross-entropy. The other parameters used are the same as Zhang and Wallace (2015), and can be seen in Table 5.3.

Table 5.3: CNN Parameters

| *Activation. Func.* | *Filter size* | *Feat. map* | *Dropout Rate* | *Regul.* | *Mini Batch* | *Epoch* |
|---|---|---|---|---|---|---|
| ReLu | 3,4,5 | 100 | 0.5 | L2 | 50 | 50 |

### 5.3.2 Recurrent Neural Network (RNN)

As discussed previously in Chapter 2, RNNs have the ability to encode sequential information in their cells. This characteristic makes it suitable for some sequential data, for instance, audio, video and text. However, RNN as well as the most used RNN-based architecture, LSTM, only allows encoding information in one direction (left-to-right), which limits the amount of contextual information retained in its cells. To overcome this limitation and also to capture more contextual information of sequential data, the authors Graves and Schmidhuber (2005) proposed a new LSTM architecture called BiLSTM, where it is possible to encode both sides information (left-to-right and right-to-left). Therefore, in this chapter, we have opted to use the BiLSTM model instead of unidirectional LSTM.

Furthermore, similarly to CNN, we have used Word Embedding vector as input to the model and *softmax* function as output layer. Also, the other hyperparameters on this model are based on the optimal values found for sentence categorisation tasks presented by Reimers and Gurevych (2017). In table 5.4, we provide these hyperparameter values. The dropout is used in the recurrent units as well as output layers.

Table 5.4: BiLSTM Parameters

| *Activation. Func.* | *Dropout rate* | *Regul.* | *Recurrent Units* | *Mini Batch* | *Epoch* |
|---|---|---|---|---|---|
| ReLu | 0.5 | L2 | 100 | 32 | 50 |

### 5.3.3 Deep-learning Models Summary

In this chapter, we are using the models presented in Table 5.5. It is possible to see their characteristics, such as algorithm and feature representation. Also, for feature representation, we provide information regarding the type of pre-trained model, the source used for training and dimension size. Moreover, we use the ID in column *Model* to identify the models throughout this chapter.

Table 5.5: Summary Configuration.

| *Model* | *Algorithm* | *Feat. Representation.* | *Pre-train. model* | *Source* | *Dim. size* |
|---|---|---|---|---|---|
| **MDL-1** | SVM | WEMB | Word2vec | Google News | 300 |
| **MDL-2** | SVM | WEMB | Glove | Common Crawl | 300 |
| **MDL-3** | SVM | SEMB | Sent2vec | SNLI | 600 |
| **MDL-4** | SVM | SEMB | InferSent | Wikipedia | 4096 |
| **MDL-5** | SVM | SEMB | SentEncoder | Wiki, Web News, SNLI | 512 |
| **MDL-6** | CNN | WEMB | Glove | Common Crawl | 300 |
| **MDL-7** | CNN | WEMB | Word2vec | Google News | 300 |
| **MDL-8** | RNN | WEMB | Glove | Common Crawl | 300 |
| **MDL-9** | RNN | WEMB | Word2vec | Google News | 300 |

## 5.4 Baseline Models (Extended Experiment)

In this section, we present the results of the baseline classification models used for detecting software vulnerability communication using 5 datasets. These models use SVM algorithm combined with traditional feature representation (BoW, *word n-grams* (1,4) and *char n-gram* (1,4)). In addition to the previous experiment shown in Chapter 4, we have included two more hacker forum datasets to complement the results. The results presented in Table 5.6 shows that BoW and *char n-gram* provide comparable performance, recording, on average, 0.75 and 0.55 of Average Class Accuracy and Positive Recall. Also, it shows that *word n-grams* feature representation achieved the poorest performance, with 0.68 and 0.38 in the same metrics. This experiment emphasises that BoW and *char n-gram* are the best choice among traditional classification models. However, considering computational performance, BoW is more efficient than *char n-grams* due to the need of less features. As a result, we are using SVM+BoW model as a baseline for the following experiments in Section 5.5.

Table 5.6: Baseline Results Using SVM and Traditional Feature Representation

| | Metric | Bag-of-Words | Word n-gram(1,4) | Char n-gram(1,4) |
|---|---|---|---|---|
| **D1** | Avg. acc | 0.70 | 0.64 | **0.71** |
| | Pos. recall | 0.45 | 0.32 | **0.47** |
| **D2** | Avg. acc | **0.88** | 0.78 | 0.86 |
| | Pos. recall | **0.78** | 0.57 | 0.76 |
| **D3** | Avg. acc | 0.75 | 0.71 | **0.76** |
| | Pos. recall | 0.57 | 0.47 | **0.60** |
| **D4** | Avg. acc | **0.70** | 0.64 | 0.68 |
| | Pos. recall | **0.46** | 0.32 | 0.44 |
| **D5** | Avg. acc | 0.74 | 0.64 | **0.75** |
| | Pos. recall | 0.50 | 0.30 | **0.52** |
| **Avg** | Avg. acc | **0.75** | 0.68 | **0.75** |
| | Pos. recall | **0.55** | 0.39 | **0.55** |

## 5.5 Analysis of WEMB as Feature Representation

This section is focused on answer the following: (1) Whether robust WEMB language models provide better accuracy compared to the traditional BoW as feature representation in detecting software vulnerability communication, and (2) Whether the security specialised WEMB is most suited for this task, compared to a common (not specialised vocabulary) WEMB model.

To answer the first question, we have used WEMB models pre-trained with general-purpose vocabulary extracted from a range of sources, such as Google News, Wikipedia, Twitter and Internet sites. We use these models in combination with SVM algorithm to create the classification model. Afterwards, we compare the achieved results with our baseline model, SVM+BoW.

To answer the second question, we have used a WEMB model trained in a security-focused vocabulary. The source used for training is a well-known security database of vulnerability descriptions maintained by the National Institute of Standards and Technology (NIST). Afterwards, we compare the performance of both classification models, security and general-purpose vocabulary.

To provide a thorough analysis of these language models, we have considered some characteristics, for instance, dimension, the number of tokens trained, volume of vocabulary and proportion of overlapping words.

### 5.5.1 Pre-trained WEMB Models: General-purpose Vocabulary

As discussed in Section 5.2.1, one of the investigated models is the *word2vec*, introduced by Mikolov et al. (2013), which is based on a three-layer neural network. We are using the pre-trained *Skip-gram* model, publicly available in `https://code.google.com/archive/p/word2vec/`. Another popular WEMB model investigated is *Glove*, introduced by Pennington et al. (2014), differently from the previous model, this is based on the factorisation of a word co-occurrence matrix. The pre-trained *Glove* model is available in `https://nlp.stanford.edu/projects/glove/`.

Both models are created with general-purpose vocabulary. As seen in Table 5.7, *word2vec skip-gram* (SG) uses Google News, while the *Glove* (G1) uses Wikipedia, (G2) uses Internet content and (G3) uses Twitter as corpus. In addition, they vary in dimension size (ranging from 200 to 300), training size (ranging from 6 billion to 100 billion of tokens), and size of vocabulary (400 thousand words to 3 million words). Each of these models is identified by the names on ID column for the remainder of this section.

We have analysed the suitability of these models in a downstream classification task

Table 5.7: WEMB Pre-Trained Models Description

| *ID* | *Type* | *Source* | *Dimension* | *Trained size* | *Vocab. size* |
|---|---|---|---|---|---|
| **SG** | SkipGram[1] | Google News | 300 | 100B | 3M |
| **G1** | Glove | Wikipedia 2014 | 300 | 6B | 400K |
| **G2** | Glove | Common Crawl | 300 | 42B | 1,9M |
| **G3** | Glove | Twitter | 200 | 27B | 1,2M |

[1] Word2vec

to detect software vulnerabilities communication. As mentioned in Section 5.2.1, for representing the entire messages, we are averaging the WEMB values of each word within the sentence to a unique vector dense method. The results of these models using WEMB are provided in comparison with the baseline model (SVM+BOW) in Figure 5.5. It is seen that all models using WEMB do not provide improvement compared to the baseline. Furthermore, comparing the use of WEMB language models, we see that the most suitable for this task is the *Glove* language model. Also, among all different *Glove* language models, SVM+G2 recorded the best results with the mean Avg. Class Accuracy of 0.64, followed by SVM+G1 and SVM+G3, with 0.62 and 0.61 respectively. This result indicates that general-vocabulary WEMB model is not better than traditional feature representation, for instance, BoW.

Additionally, the results suggest that the training size (number of tokens), as well as the size of vocabulary, has an influence on the performance of the classification model using WEMB as feature representation. As seen in Table 5.7, the best WEMB model, G2, which has achieved the best results, is also the biggest model in terms of number of tokens and vocabulary size compared to the others.

Another factor that affects the results is the proportion of *Out-of-Vocabulary* (OoV) words, that is, the proportion of words that lack embedding representation due to specificity of vocabulary (rare words) (Won and Lee, 2018). In Table 5.8, we present the overlapping proportion of words each dataset (D1, D2 and D3) has in common with the pre-trained language models (SG, G1, G2 and G3). By this table, we see that

D2 has the larger proportion of overlapping words compared to D1 and D3. As a result, the model created with this dataset has provided the best individual performance, followed by D3 and D1, which has a lower proportion of overlapping messages and lower performance respectively. In other words, the higher the proportion of overlapping messages between dataset and WEMB model, the better the performance achieved by the classification model.

Table 5.8: Proportion of Words For Each Dataset Within Pre-Trained WEMB Models

| *ID* | *D1* | *D2* | *D3* |
|---|---|---|---|
| **SG** | 57% | 81% | 74% |
| **G1** | 62% | 85% | 80% |
| **G2** | 75% | 92% | 89% |
| **G3** | 61% | 83% | 76% |

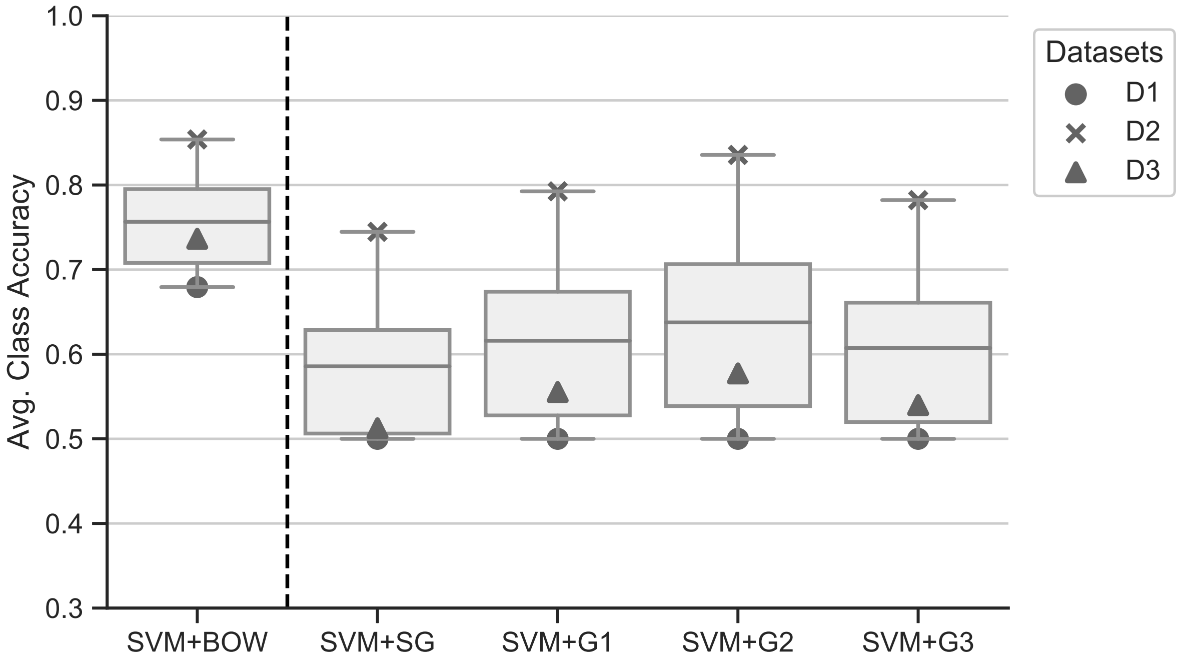


Figure 5.5: SVM using Bag-of-words (BoW) and Word Embedding (SG, G1, G2, G3) as Feature Representation

### 5.5.2 Pre-trained WEMB Models: Security Vocabulary

In this section, we analyse the use of security-specific vocabulary for training the language models. For doing this, we have trained a Word2Vec model with a security corpus collected from National Vulnerability Database (NVD). This database is a catalogue of information related to real software vulnerabilities. In this catalogue, different types of information are available, such as the description of each vulnerability, a list of affected software, the severity (Low, Medium, High and Critical), software vendors and others. From these datasets, we have used only the vulnerability descriptions disclosed over 10-year period to train our language model, representing 102,292 descriptions from 2009 to 2019. The total number of tokens within this corpus is 3,444,195, which have resulted in the total size of the vocabulary of 64,765 words.

In Table 5.9, we present a comparison of the overlapping proportion between the dataset and the language models created with security-specific and common-specific vocabulary. Also, we provide some characteristics as the number of tokens for training, the size of vocabulary and dimension per model.

Table 5.9: Comparing Language Models Trained With Different Vocabulary (General-x Security-focused)

| *Corpus* | *Word in common (%)* | | | *Train Size* | *Vocab. Size* | *Dim* |
|---|---|---|---|---|---|---|
| | *D1* | *D2* | *D3* | | | |
| **General** | 57% | 81% | 74% | 100B | 3M | 300 |
| **Security** | 34% | 57% | 40% | 3M | 64K | 300 |

We observed that, in general, the datasets D1, D2, D3 have fewer words in common with the language model trained with security-specific vocabulary compared to the common-specific vocabulary. Moreover, it is noted that a considerable difference exists regarding the number of tokens each model was trained with. One is trained with 3M tokens and the other with 100B. As a result, the size of the vocabulary of the security-vocabulary model is 64k, whereas the general-vocabulary is far more, with

3M words.

However, despite the security model being smaller than the general-specific, the classification performance of both models achieved very similar results. As seen in Figure 5.6, the models trained with the security-vocabulary recorded the best Avg. Class Accuracy and Recall in 2 of 3 datasets (D2, D3), which suggests that the use of language models with security-vocabulary corpus has a positive influence on the results. These models provided comparable results with general-vocabulary models using fewer tokens for training and also having fewer words in common.

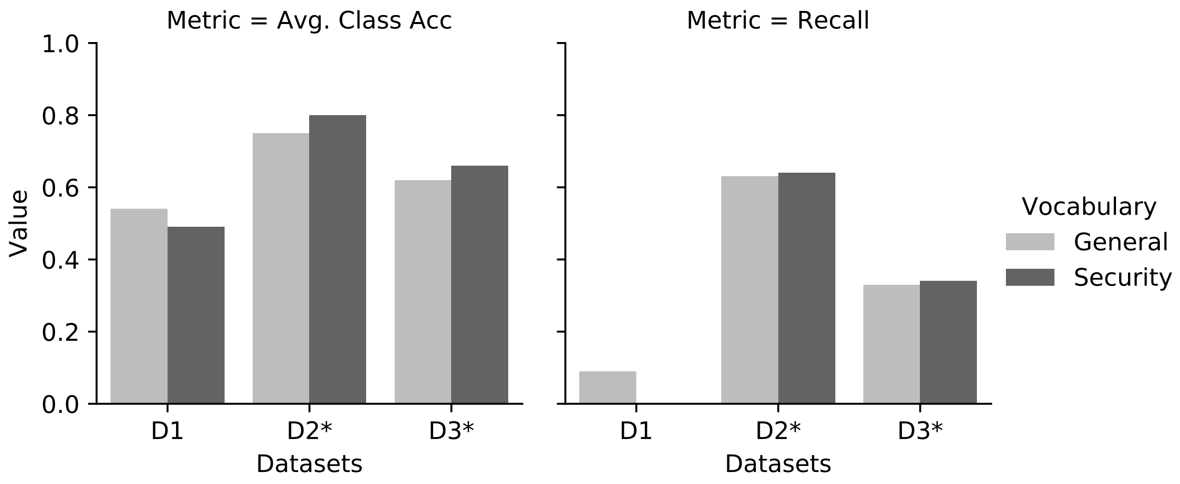


Figure 5.6: Classification Performance General x Security Vocabulary Models

## 5.6 Analysis of SEMB Models as Feature Representation

As previously mentioned in section 5.2.2, SEMB models improved over WEMB with regards to the word-order limitation. However, it is not known whether this improvement is translated into better classification accuracy on classification models for detecting software vulnerability communication. As a result, in this section, we investigate the performance of models using SEMB as feature representation in our specific classification task. Furthermore, we compare a range of different combinations of models, grouped in the following categories: SVM+WEMB (MDL-1 and MDL-2) and SVM+SEMB (MDL-3, MDL-4 and MDL-5). The results of these models, in terms of average class accuracy and positive recall, can be seen in Table 5.10 and the details of each individual model configuration is seen in Table 5.5.

Analysing the result of this experiment, we have observed a very large difference between SVM+SEMB and SVM+WEMB performance. On average, the best WEMB model does not outperform the worst SEMB model. Additionally, WEMB presented a poor recall performance of 0.23 and 0.33 for MDL-1 and MDL-2.

Considering only the SEMB models MDL-3 (*Sent2vec*), MDL-4 (*InferSent*) and MDL-5 (*SentEncoder*), we see that MDL-5 has achieved the best result, with 0.82 and 0.74 of average class accuracy and positive recall respectively. It should also be noted that the second-best is MDL-3, with 0.75 and 0.60 of average class accuracy and positive recall.

The results suggest that SEMB are better models to use in detection of software vulnerability communication as they are more robust in terms of sentence representation. With regards to the best results, MDL-5, the main difference compared to the other SEMB models investigated is that this model uses a multi-task architecture (*SentEnconder*), which uses unsupervised and supervised approaches to create the final embedding vector.

Table 5.10: Experimental Results with Best Metric in Bold per Category and per Dataset

| | | **SVM+ Word. Emb.** | | **SVM+ Sent. Emb** | | |
|---|---|---|---|---|---|---|
| | | MDL-1 | MDL-2 | MDL-3 | MDL-4 | MDL-5 |
| **D1** | Avg. acc | 0.54 | **0.55** | 0.68 | 0.68 | **0.81** |
| | Pos. recall | 0.09 | **0.10** | 0.50 | 0.41 | **0.72** |
| **D2** | Avg. acc | 0.75 | **0.80** | 0.80 | 0.86 | **0.88** |
| | Pos. recall | 0.63 | **0.73** | 0.70 | 0.75 | **0.87** |
| **D3** | Avg. acc | 0.62 | **0.65** | 0.81 | 0.78 | **0.83** |
| | Pos. recall | 0.33 | **0.45** | 0.71 | 0.63 | **0.78** |
| **D4** | Avg. acc | 0.54 | **0.62** | 0.71 | 0.73 | **0.77** |
| | Pos. recall | 0.14 | **0.38** | 0.57 | 0.52 | **0.70** |
| **D5** | Avg. acc | 0.49 | 0.49 | 0.73 | 0.73 | **0.81** |
| | Pos. recall | 0.00 | 0.00 | 0.53 | 0.49 | **0.66** |
| **Avg** | Avg. acc | 0.58 | **0.62** | 0.75 | 0.75 | **0.82** |
| | Pos. recall | 0.23 | **0.33** | 0.60 | 0.55 | **0.74** |

## 5.7 Analysis CNN and RNN as Classification Algorithm

Until now, we have been using linear algorithms classifiers, such as the linear SVM and Naïve Bayes. These algorithms are known for mapping the data features space values into a linear decision boundary (Hastie et al., 2009). However, this type of algorithm does not always provide the best prediction performance, especially when the prediction output is characterised by non-linear patterns. As a result, in this section, we perform experiments using deep learning-based algorithms as classifiers, as they provide state-of-the-art results when dealing with non-linear data. With this experiment, we want to investigate whether the use of deep learning algorithms for our specific task improves the detection performance compared to the best traditional model seen so far (MDL-5).

As a result, we selected CNN and RNN as they have been widely used in text classification as well as other NLP tasks (Gan et al., 2016, He et al., 2015, Hochreiter and Schmidhuber, 1997, LeCun et al., 1990). The architectural details of these algorithms are provided in Section 5.3.1 (CNN) and Section 5.3.2 (RNN). In Table 5.5, we provide identification and characteristic of the model built with these algorithms. For models built with CNN, the identifiers are MDL-6 and MDL-7, using *Glove* and *word2vec* as feature representation. For models built with RNN, the identifiers are MDL-8 and MDL-9, also using *Glove* and *word2vec* respectively.

In Table 5.11, we provide the results of these models. We see the best results so far in detecting software vulnerability communication compared to the previous results in Section 5.6. The performance of the model MDL-6, MDL-7, MDL-8 and MDL-9, which uses DL-based algorithms as classifier overcomes the best previous model, MDL-5 (SVM+*SentEncoder*). Comparing only the CNN- and RNN- based models, we see that MDL-7 (CNN + *word2vec*) achieved the best results, with 0.96 of average class accuracy and 0.93 of positive recall, in average, whereas MDL-8 (RNN + *Glove*), the second-best model, achieved 0.84 and 0.77 in the same metrics respectively. In terms of feature representation, we have observed only a small difference in performance either by using Word2Vec or Glove, which suggests that both language models are equally suitable for the task. Another point to mention is that MDL-8 (the second-best model overall) does not improve with great extent compared to the previous model, MDL-5 (SVM+*SentEncoder*). The former achieves 0.74 and the latter 0.77 of recall. However, the training time of MDL-8 (RNN) is much longer than MDL-5 (SVM), which might be seen as a bottleneck in a real-life situation depending on the size of the training and computational resource available.

Table 5.11: Experimental Results CNN and RNN Based Models)

| | | **CNN+ Word Emb.** | | **RNN+ Word Emb** | |
|---|---|---|---|---|---|
| | | MDL-6 | MDL-7 | MDL-8 | MDL-9 |
| **D1** | Avg. acc | **0.95** | **0.95** | 0.80 | 0.82 |
| | Pos. recall | **0.91** | **0.91** | 0.74 | 0.75 |
| **D2** | Avg. acc | **0.98** | 0.98 | 0.94 | 0.90 |
| | Pos. recall | **0.97** | 0.96 | 0.84 | 0.81 |
| **D3** | Avg. acc | **0.95** | **0.95** | 0.82 | 0.81 |
| | Pos. recall | **0.92** | **0.92** | 0.72 | 0.71 |
| **D4** | Avg. acc | 0.96 | **0.97** | 0.83 | 0.81 |
| | Pos. recall | 0.92 | **0.95** | 0.79 | 0.74 |
| **D5** | Avg. acc | 0.95 | **0.95** | 0.84 | 0.82 |
| | Pos. recall | 0.90 | **0.91** | 0.77 | 0.76 |
| **Avg** | Avg. acc | 0.95 | **0.96** | 0.84 | 0.83 |
| | Pos. recall | 0.92 | **0.93** | 0.77 | 0.75 |

## 5.8 Discussion

Comparing the results of the experiment (Section 5.6) to the baseline (Section 5.4), we notice that replacing classical feature representation with the semantically richer WEMB language models does not result in better classification performance. This is similar to Zhang et al. (2015), which WEMB has not improved over traditional classifiers (linear algorithm + n-grams). Also, in a sentiment analysis task, as shown in Liu (2017), and spam detection in SMS messages, presented by Lee and Kang (2019), WEMB models have not provided any improvement over traditional classifiers. These results highlight that the accuracy of machine learning models is tied in with finding an appropriate combination of algorithms and input features for the specific task Perone et al. (2018), in which the experimentation and evaluation are the tools for achieving the best results.

With regards to SEMB models, we have noticed an improvement in detection of hacker communication compared the traditional and WEMB models. As we previously discussed, traditional representations such as BoW, *n-grams*, and even distributed representations, such as *word2vec* and *Glove* do not consider the order in which a word occurs in a sentence, whereas all SEMB models do inherit this property (See the comparison in Table 3.9). Therefore, the results achieved suggest that, in downstream classification tasks, this property is the key attribute that provides more accurate models for our specific task.

## 5.9 Summary and Conclusion

Detection of potential cyber threat communications in hacker forums and social media is a difficult task due to the technical vocabulary and ambiguity of certain posts. In this research, we performed an experiment with different configurations of classification algorithms and language models for detecting malicious messages related to software vulnerabilities in forums and social media. We have evaluated these models through 5 different labelled datasets (D1 to D5). Taking this into account, it has been concluded:

(1) SEMB feature representation is the best embedding for improving classification performance on linear SVM model compared to WEMB and other traditional representations, such as BoW, *n-gram and char n-gram.* In this work, 2 out of 3 models with SEMB were able to overcome the best baseline models. MDL-3 (Sent2vec) and MDL-5 (SentEconder), achieved 60% and 74% of recall, respectively, and MDL-4 (InferSent) recorded the same performance as the baseline, 55% of recall.

(2) In terms of pre-trained WEMB as feature representation, we observed that (1) they are not better than traditional feature representation, such as BoW, *n-gram and char n-gram*, (2) it was noticed there are two factors that influence the performance of WEMB in a downstream classification task, one is the size of vocabulary and other is the di-

mension size. (3) Security-vocabulary pre-trained WEMB provided comparable results with General-vocabulary WEMB, although trained in much fewer data. These results suggest that a WEMB trained in a security context can outperform General pre-trained embedding if trained on similar dataset size.

(3) In terms of classification performance overall models, we found that the best model, MDL-7 (CNN + WEMB (*word2vec*)) can achieve 93% of positive recall and 96% of average class accuracy and MDL-8 (RNN + WEMB (*Glove*)), the second-best model, can achieve 0.77 of positive recall and 0.84 of average class accuracy. However, the training time of the RNN model is much superior to CNN, which can be a bottleneck for a real-life situation depending on the size of the training data. We believe that this configuration shows promise as an optimal approach for detecting posts related to software vulnerabilities. In practice, these models can be used by companies for prioritising security updates (patching) of vulnerable systems in their assets.

**CHAPTER 6**

# Effects of Concept Drift on Classification Models Applied to Software Vulnerability Communication

In recent times, machine learning algorithms have been popularised due to their capacity for solving real-world problems through the discovery of useful patterns in data. These algorithms have revolutionised tasks such as image recognition (e.g., handwriting and face classification) and natural language understanding (e.g., sentiment analysis, Named-entity recognition). The enthusiasm surrounding data-driven learning approaches has expanded to a wide range of domains, including cyber security (Greengard, 2016).

In general, machine learning models applied to cyber security tasks are designed with the assumption of a static world view, which states that the model trained with current data will be useful for predicting long time future data. Such an assumption does not accommodate dynamic changes of concept in certain problems over long periods (Sethi and Kantardzic, 2018). As a result, the accuracy of these models can be affected, which render them less effective or useless as time goes on. This phenomenon is known as Concept drift and it is commonly seen in natural language tasks where the language usage and meaning change constantly (Guerra et al., 2014, Wang et al., 2013).

In cyber security initiatives that use hacker communications, variations in the concept are observed when new forms of malicious hacking emerge and are shared in social media channels. Supporting this, the authors in Zhao et al. (2016) provided a study on different hacker jargon used to express well-known security issues in hacker forums. These forms of expression are not static and are susceptible to changes over time. Also, as seen in Chatterjee and Thekdi (2020), between 2013 to 2017, there was an increase in new types of vulnerabilities in software that were not previously categorised according to a stabilised taxonomy. This implies that hackers are communicating this information by using different words and grammatical structures for expressing new concepts. As a consequence, in this dynamic context cyber defenders are trying to operate to protect computer systems from cyber threats.

Followed by this discussion, our aim in this chapter is to investigate the extent and effect of Concept Drift on the accuracy of Machine Learning (ML) models used to detect software vulnerabilities mentioned in social media posts. We base our experiment in the following research questions:

- Does the accuracy of our machine learning models, created to detect software vulnerabilities communication in social media posts, degrade over time, indicating Concept Drift?
- Can we alleviate this problem using a typical Concept Drift solution method?

The structure of this chapter is as follows: In Section 6.1, we review a selection of related works on Concept Drift, some are applied to the security domain. In Section 6.2, we describe the experimental approach, including dataset and static/dynamic evaluation methodology used to measure the performance of the model throughout a period. In Section 6.3, we provide an analysis of the performance of the model in the 1 year using Twitter (Surface Web) and hacker forum (Deep Web) datasets. In Section 6.4, we measure the slope of the line to identify the best approach to contain the deterioration of performance in these models. Finally, in Section 6.4, we conclude presenting the contributions of this chapter.

The experiments in this chapter were published in:

- Queiroz, A. L., Keegan, B., and Mckeever, S. (2020). Moving Targets: Addressing Concept Drift in Supervised Models for Hacker Communication Detection. In *International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–7.

## 6.1 Concept Drift and Security Research

The field of Concept Drift is an entire research domain with studies focused on providing forms of avoiding performance decrease as a result of changes in the concept. These are usually focused on triggering the retraining of models when changes in the underlying distribution of the incoming data are detected. For instance, in Bifet and Gavaldà (2009), the authors proposed an algorithm that observes the change in the distribution of the data using a dynamic sliding window. Furthermore, an algorithm proposed by Ross et al. (2012) detects the drift in an on-line manner without the need for storing data points in memory, which provides a more efficient algorithm in terms of computational resource.

However, recent work with a similar purpose as this research (hacker communication detection) has shown little or no concern with the evolving nature of communications amongst peers in social media platforms. Furthermore, in the security domain, more specifically in malware analysis, a study has identified that the performance of machine learning models is affected by the rapidly evolving (drift) of malware software, as a result, further unseen malware is not identified by models trained in a long time past data (Jordaney et al., 2017). Moreover, this study has highlighted that computer security is a dynamic problem, with new forms of attack rising rapidly and changing over time.

## 6.2 Experimental Approach

The experiment presented here is guided by two principal goals: The first is to analyse whether machine learning models applied to this research purpose degrade over time. Such an effect is indicative of Concept Drift. The second is to analyse a method to alleviate this problem, avoiding performance degradation and allowing a more time-resistant model.

To achieve these goals, this experiment has focused on evaluating the temporal changes to investigate the robustness of machine learning models applied to detect vulnerability communication. Thus, to visualise eventual changes in performance, the models are trained and tested in an organised manner, which considers the date and time that the messages were posted on social media platforms. By this, we are also providing a simulation of the use of these classification models in a real-life situation.

Furthermore, we have selected datasets from different sources (D1, D2 and D4), which consists of user posts gathered from Surface Web (Twitter) and Deep Web (hacker forums). Additionally, these datasets were selected to achieve a minimum of 10% (approximately) proportion of positive instances within the first months of the year (first 4 months), which leaves us with sufficient time for evaluating the following 8 months data (total 1-year period). Hence, we can apply sampling and reduction techniques seen in Chapter 4, including random sampling to increase the performance of the start model as done in Queiroz et al. (2019).

The importance of evaluating accordingly is to investigate how long the model would retain performance after its creation - in other words, to measure the robustness of the model against the changes in the concept. Finally, in the following subsections, we describe in details the steps for performing this experiment, i.e., how the labels of the dataset were handled and the methodology, which includes how we have performed the Prequential Evaluation using Monthly Time Window (MTW).

### 6.2.1 Transforming into a Binary Classification Task

In this experiment, we have followed the same binary approach we discussed in Chapter 4, as it has shown similar results to the multi-class approach. Also, from a security standpoint, it is prudent to capture the *Undecided* messages as a potential malicious problem. Therefore, we have transformed this task into binary classification by placing the *Undecided* and *Yes* messages into the positive class, whereas *No* messages are the negative class. Further details on the description of the dataset used can be referred back to 3.2 and the performed labelling task is found in Section 3.3.3.

It can be observed that all datasets have an imbalanced number of instances of positive and negative classes, with the minority proportion of these messages representing the positive, thus, software vulnerability communication. In Table 6.1, the proportion of instances within the classes, as well as respect to the number of posts and the average number of words is shown.

To compare these experiments fairly we choose to use a 1-year peak for each, as the shortest dataset has approximately a 1 year range.

Table 6.1: Dataset Description

| *ID* | *Source* | *Type* | *No. of inst.* | *Distrib. (pos/neg)* | *Avg. words* |
|---|---|---|---|---|---|
| **D1** | Hacker Forum | Deep web | 1,682 | 10/90% | 50 |
| **D2** | Twitter | Surface web | 1,927 | 15/85% | 13 |
| **D4** | Hacker Forum | Deep web | 1,966 | 13/87% | 78 |

### 6.2.2 Methodology

The methodology for creation and evaluation of models is based on the timestamp of the messages, where the training instances (used to create the model) occurs in a period before the testing instances (used to evaluate the model) as seen in Figure 6.1. Each evaluation is given by a Monthly Time Window (MTW) in a Predictive Sequential

Evaluation (Prequential). In this approach, we provide the accuracy of the model for each MTW, such that its performance is tracked over time.

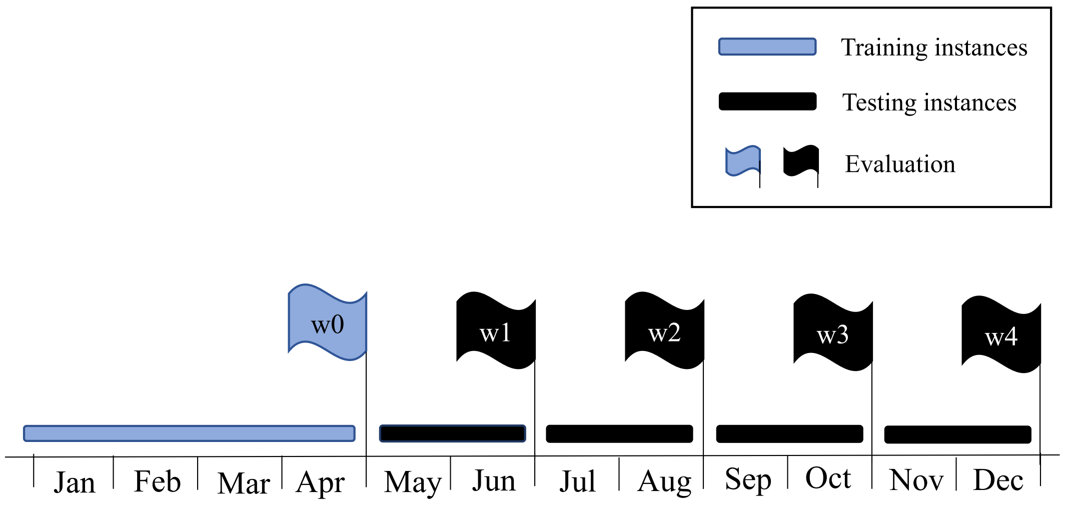


Figure 6.1: Monthly Time Window (MTW) Instances for Training/Testing the Model in 1-year Period, and Evaluation Checkpoints at $w0, w1, w2, w3, w4$

### Prequential Evaluation

Considering the changing nature of hacker language with the emergence of new forms of exploiting software vulnerabilities, we hypothesise that models trained in past instances will not present the same performance in new, or future, instances due to the Concept Drift. For this reason, we propose to simulate the use of these models in real-life situations, where the model is trained and evaluated with consideration of the time when the messages occur in the social media platform. The data used as input to the model is therefore organised by date of publication. For this streaming task, we are using Sliding Window prequential evaluation similar to Hidalgo et al. (2019), which in our case the windows are fixed by months of the year.

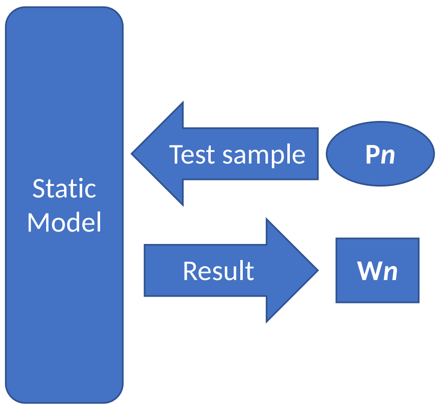
**Monthly Time Window**

To evaluate the performance of the model over 1-year period, we selected the use of a monthly time window to train and test the model. These time windows are organised for every 2 months (bimester), although, for the training phase of the experiment, we have used 2 bimesters (4 months) due to the need of a minimum number of positive instances for applying oversampling technique as done in Queiroz et al. (2019). This technique provides a performance enhancement of models built using imbalanced data. For the remainder of this work, the notations shown on the graphs are as follows: $w_0$, is the start model, which is trained and cross-validated in 2 bimesters messages, whereas the time window $w_1$, $w_2$, $w_3$, $w_4$ are the results of the test phase with the incoming messages with messages of one bimester each.

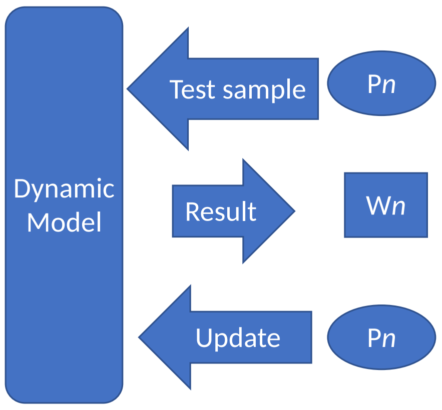
## 6.3 Analysing the Concept Drift in a 1-year Period

In this section, we firstly describe the creation (training) of the start model, which is the baseline for the following two evaluation approaches: Static and Dynamic. The main difference regarding these two approaches is that, in Static, the model is not re-trained during the evaluation period of 8 months, while in the Dynamic approach, the model is re-trained with new instances of each month after its evaluation. These two approaches are better exemplified in Figure 6.2 and details are described in Section 6.3.2 and 6.3.3, respectively.

Furthermore, for creating the models, we used Support Vector Machine (SVM) with linear kernel, and Bag-of-words (BoW) as feature representation. This combination was used in previous experiments (Chapter 4) and has provided reasonable computational performance and accuracy. Additionally, we choose not to *tune* the hyperparameters of SVM algorithm (e.g., parameter C) as these values are data-dependent, which means that the optimal hyperparameters can be used only for the applied dataset. As a result, to fairly compare the models' performance in all datasets, we have used



(a) Static Evaluation



(b) Dynamic Evaluation

Figure 6.2: Experimental Procedure, Where P = sample, and W = result in Avg acc. and $n$ = period

the same hyperparameters values for all models, which are default values of SVM in Sci-kit 0.20.2v.

Finally, in Table 6.3, we see a detailed description of each phase of the experiment. This table has also provided information regarding the number of instances in each phase, the proportion in each class, and the phases of the evaluation.

### 6.3.1 Start model (Training Phase)

All experiments presented in this paper begin with the start model, with its performance reported in the figures of this chapter as $w0$. For training these models, we have used imbalanced datasets which contains a few instances of positive class (approximately 10%). As mentioned in Section 4.5, this is not the ideal proportion for creating a classifier, which might affect the performance of the model negatively (Ganganwar, 2012). To provide a better condition, we have increased the actual number of positive instances by using random oversampling technique, as shown in Section 4.6. This technique is also used to enhance the accuracy of models.

Furthermore, we evaluate the performance of the start model using a 10-fold-cross validation. The values recorded for the hacker forums datasets (D1 and D4) are 67%

and 68%, respectively, and Twitter dataset (D2) is 77%, seen in Table 6.2. We highlight that the sampling was applied only in training folds of the cross-validation, leaving the test folds without oversampling. The average results of each fold are reported in w0. This sampling represented an increase of the positive instances, which achieved 33% of the dataset, as seen in Table 6.3

As seen in Table 6.3, the static model ($w0$) achieved 33% positive and 67% negative instances after sampling. However, the random sampling is not applied into test ($w1, w2, w3, w4$) to avoid a biased result. Instead, each test time-window persists with the exact number of instances that correspond to that period.

Table 6.2: Start Model Avg. Class Accuracy per Dataset

| | SVM+BOW |
|---|---|
| D1 | 0.67 |
| D2 | 0.77 |
| D4 | 0.68 |

### 6.3.2 Static Model Evaluation

In this section, we aim to answer the first question stated in the introduction of this chapter: "Does the accuracy of machine learning models, created to detect software vulnerabilities communication in social media posts degrade over time?". To answer this, we have tested the performance of the start model ($w0$) over a sequence of unseen instances ($w1, w2, w3, w4$). These instances are organised accordingly with the time they were posted on social media. Also, the start model is not re-trained during the test. The performance recorded for $w0$ is given by the results of the Static Model trained in 4 month period, whereas for $w1, w2, w3, w4$ are provided by the evaluation over instances of 2-months (bimester) period, with a total of 1-year period.

The performance evaluation is presented per each dataset (D1, D2, D4). Moreover, we have plotted the performance of each model along with a regression line that best

Table 6.3: Distribution of Instances in Each Phase

| | | | D1 | | D2 | | D4 | |
|---|---|---|---|---|---|---|---|---|
| Phase | Periods (months) | Window | No. Msgs | Distrib. % (pos/neg) | No. Msgs | Distrib. % (pos/neg) | No. Msgs | Distrib. % (pos/neg) |
| Start Model[1] | 1st to 4th | w0 | 342 | (33/67) | 826 | (39/61) | 314 | (44/56) |
| Test | 5th and 6th | w1 | 144 | (10/90) | 317 | (11/89) | 179 | (16/84) |
| Test | 7th and 8th | w2 | 149 | (10/90) | 263 | (13/87) | 89 | (20/80) |
| Test | 9th and 10th | w3 | 185 | (12/88) | 291 | (23/77) | 106 | (7/93) |
| Test | 11th and 12th | w4 | 150 | (9/91) | 257 | (19/81) | 207 | (8/92) |

[1] Positive class instances in training phase are oversampled by 300% according to Queiroz et al. (2019)

fits these data points over time. This line also represents the rise over run ratio of the model in which the slope is calculated and shown in Section 6.4. The x-axis represents the time window and the y-axis is the performance of the model in average class accuracy for that period. According to Figure 6.3, we observe an apparent drift occurring only in both hacker forums datasets (D1 and D4), which is seen by a negative tendency line. Whereas, for D2 (Twitter), we see no apparent deterioration of the performance with an almost flat tendency line.

These results suggest that models created with hacker forums source (D1 and D4) are more sensitive to Concept Drift. This effect is perceived by the performance decrease in models in less than 1-year period. Additionally, through inspection of the performance values in 1-year period, it is seen that D1 starts at 61% and ends in w4 with 59%, whereas D4 has recorded a bigger drift, starting at 68% and ending at 56%. Regarding the model with positive slope, D2 (Twitter), we see the start and ending performance at the same value, 77% avg. class acc, and the lowest performance is only seen $w1$, at 76%.

Finally, despite the contents in D1, D4 (hacker forums) and D2 (Twitter) are related to security and technology in general, they vary in purpose. Twitter is used to inform a broad audience (expert or not), whereas hacker forums are mainly used by computer experts, thus, more likely to contain a variate vocabulary, which includes technical language. Thus, this explains the different performance of these models.

### 6.3.3 Dynamic Model Evaluation

In this section, we aim to answer the second question stated in the introduction of this chapter: "Can we alleviate this problem using a typical Concept Drift solution method?". To answer this question, we perform a Dynamic Evaluation, where the model is periodically re-trained with incoming instances in a systematic manner in two different forms:

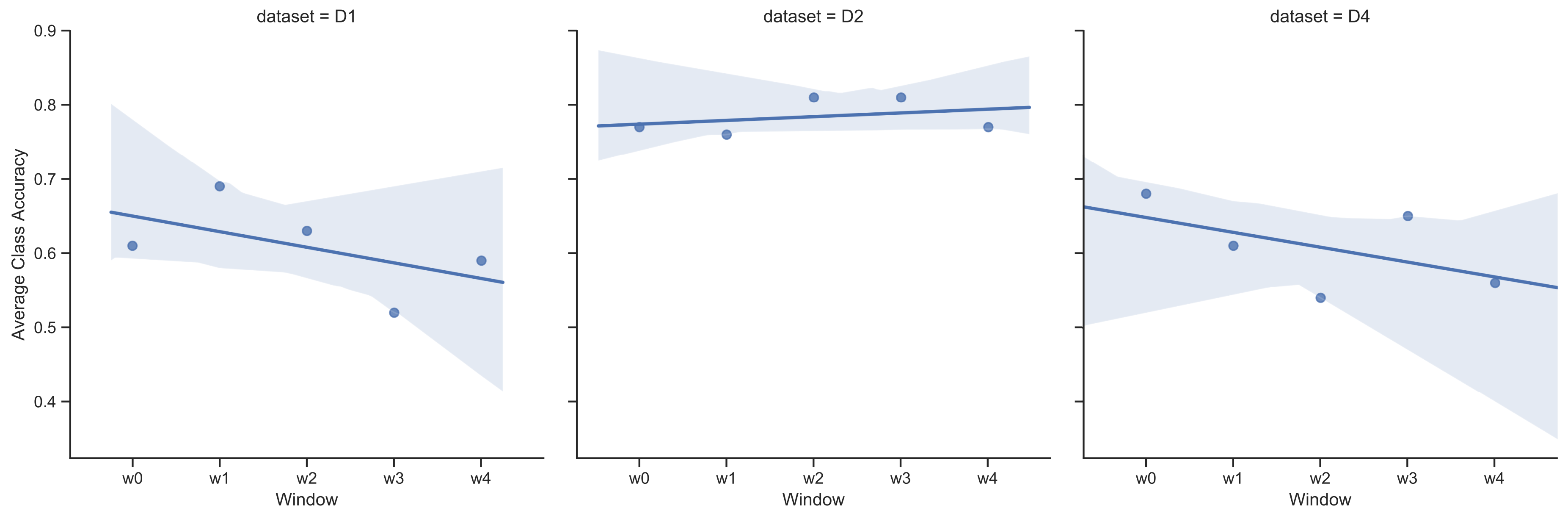


Figure 6.3: Static Model (Baseline) - Average Class Accuracy Results of the Model Trained in 4 Months Period ($w0$) and Tested in Every Two Months ($w1...w4$) in 1-year Period

- First, we investigate the optimal proportion of instances for updating the model using different proportions (25%, 50%, 75% and 100%) and compare these results with the baseline Static model.
- Second, on top of the optimal proportion found previously, we add different weights on the incoming instances. The values are in a ratio of 2:1, 3:1, 5:1 and 10:1 (newest:old). With this, we aim to understand whether this approach can be used to enhance the accuracy of the models over time. Afterwards, we compare these results against the previous Dynamic Evaluation (unweighted instances)

The following subsections provide more details and results of these two dynamic approaches.

### Dynamic Updating (unweighted instances)

In this experiment, we have performed a periodic update of the start model ($w0$) using instances from time windows $w1, w2, w3, w4$, such that the model is first tested with sample $P_n$ to provide the result for $W_n$ and is re-trained with the same $P_n$ instances in sequence, as seen in Figure 6.2 (b).

The instances are selected randomly by different proportions (25%, 50%, 75% and 100%). The criterion for defining the optimum values would be the models that need the lowest re-training proportion and achieve highest performance accuracy over time. Low re-training proportion means a reduction on the resource needs to provide the labels (expert labellers). In Figure 6.4, we see the start model retrained by a different proportion of instances and its respective results over time. The results are provided per dataset.

The first impression we have looking at this graph is that updating the model with the newest instances is effective in improving the performance over time. This is noticed by comparing the performance enhancement of all Dynamics with the Static models (thick blue line) in all re-trained proportions. This is also true in all three datasets (D1, D2, D4)

It is also worth highlighting that, updating the model with 100% of the incoming instances provides, as expected, the best results. However, reducing the update proportion to 50% of the incoming instances still achieves comparable results to 100% in the majority of the datasets. This is a fair trade-off to consider for real-life application of supervised classification models. The 50% proportion provides a significant reduction of messages to be labelled by human experts, as a consequence, it provides a reduction in time and expert people needed to perform the labelling task. Thus, we considered this as the optimal updating proportion for our applied problem.

**Dynamic Update (weighting Instances)**

In this experiment, we performed a periodic update of the start model (w0) using instances from time windows $w1, w2, w3, w4$, such that the model is first tested with sample $P_n$ to provide the result for $W_n$ and is re-trained with the same $P_n$ instances in sequence. However, in constrast to the first part, the most recent time window instances are weighted in different proportions, for instance, x2 (2:1), x3 (3:1), x5 (5:1) and x10 (10:1).

Additionally, this experiment is done on top of the optimal proportion values achieved in the last experiment (50% proportion relabelling). With this experiment, besides enhancing the performance of the model, we want to evaluate the effects of weighting the newest instances. In other words, we want to observe how important the newest instances are for the performance of the model.

Fig 6.5 shows how the model has performed after applying weights on top of the 50% of random instances. The thicker (blue) line represents the previous model retrained with 50% random instances **without** weight, whereas the thin line represents the model retrained with the same 50% random instances, **with** different proportion of weights on them x2 (2:1), x3 (3:1), x5 (5:1) and x10 (10:1). Visually, we see that the models with weighting schemes have performed best compared to the baseline

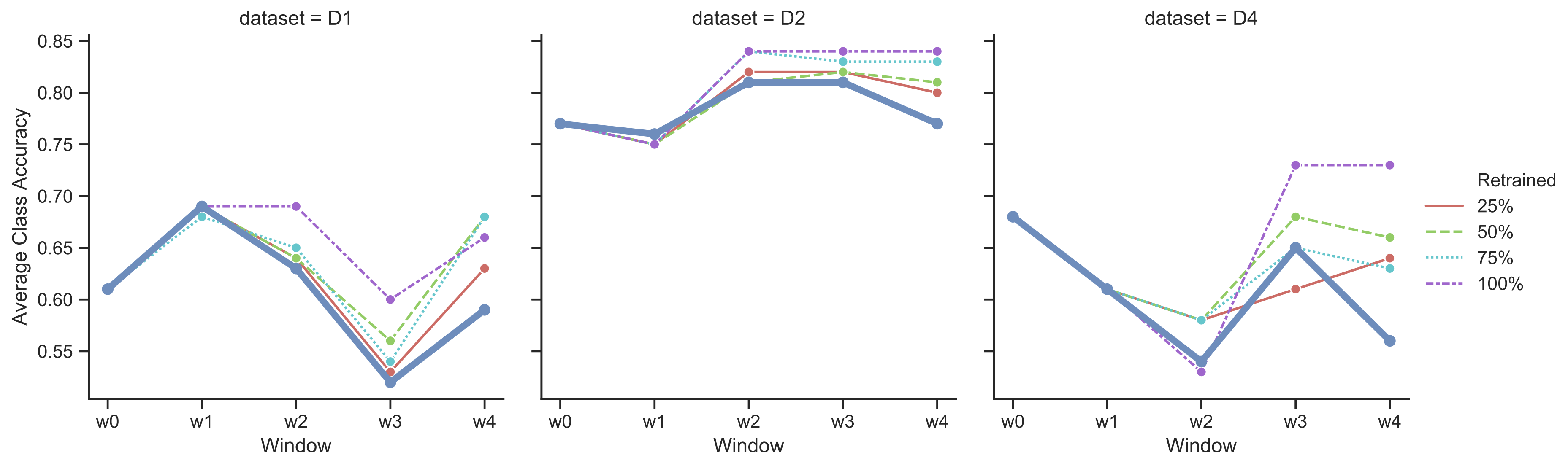


Figure 6.4: Comparing the Baseline Static Model, Bold Line (Blue), against Models Retrained With Different Proportions (25, 50, 75, 100)% of Past Instances

(thicker line).

## 6.4 Measuring the Degree of Drift of Models

In Zliobaite et al. (2016) and Kadam (2019), the authors have provided a visual explanation of four common drift patterns, they are: *Sudden*, *Incremental*, *Gradual* and *Reoccurring*. According to these concepts and by analysing the graphical performance over time in Figure 6.3, it is noted that a gradual drift in the model in D1 and D4, wherein D2 stays practically flat. However, to precisely measure this drift, we provide a calculation of the slope of the line for each dataset (D1, D2 and D4) throughout a 1-year period.

In the final set of results, negative values mean that the performance of the model points towards decreasing over time, whereas the positive values represent the opposite, pointing to an increase over time. In table 6.4 we present the slope calculation. The values for each model are presented. Also, we have used the Static model as the baseline for comparison. The dynamic models are all trained with the optimal values of 50% re-training.

As previously observed, by retraining the model with 50% of incoming instances, the performance over time is improved over the baseline static. In other words, the minimum of 50% of retraining of the incoming messages is sufficient to avoid the drift. This result strongly suggests that there is no need to provide labels to all incoming instances. The direct benefit of this results in the reduction of the labelling task workload of 50%, which also reduces the time and human resources need to perform the task.

Moreover, it was observed that when weighting the newest incoming samples, we achieved further improvements. All Dynamic models with 50% relabeled + weights (x2, x3, x5, x10) have improved over the Dynamic model without any weights. Considering the average results, the optional configuration for avoiding performance de-

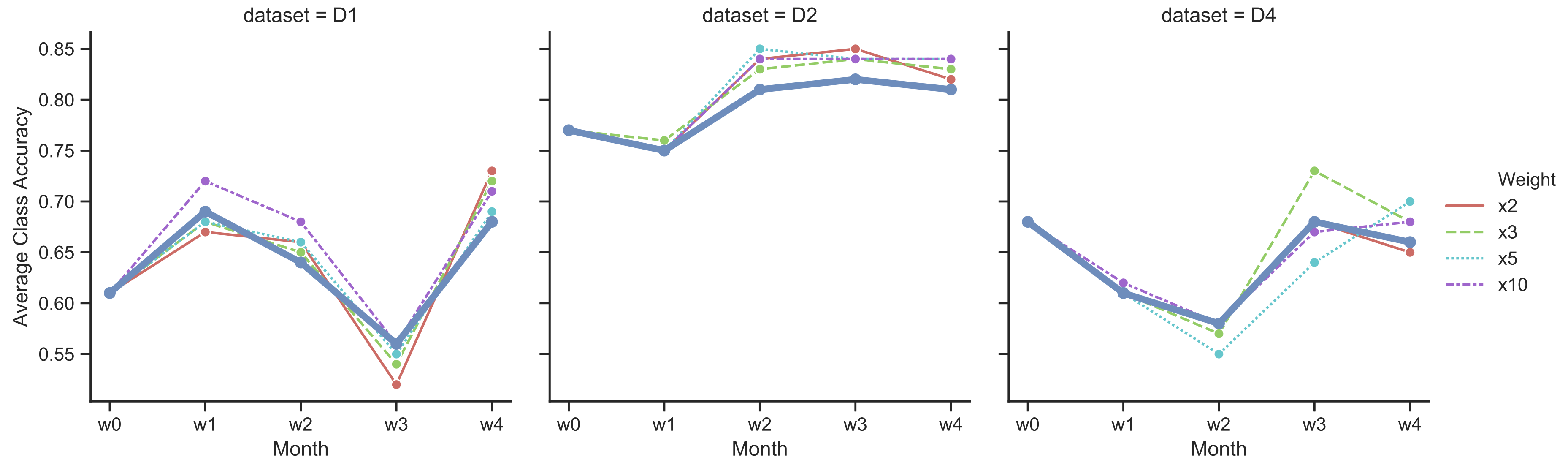


Figure 6.5: Comparing Models with Different Values of Weights on New Instances (x2, x3, x5, x10) against Model with No Weights, Bold Line (Blue). All Models Using 50% Proportion of Incoming Instances

terioration over time is given by Dyn. retraining 50% + weights on 3:1 proportion. We have also noticed that there is no need to add weights above 3:1 proportion, as this does not provide any further improvement.

Table 6.4: Slope of the Line for Static and Dynamic Models

| | D1 | D2 | D4 | Avg |
|---|---|---|---|---|
| Static. (Baseline) | -.020 | +.005 | -.019 | -.011 |
| Dyn. (50% relabeled) | +.001 | **+.015** | +.003 | .005 |
| Dyn. (50% relabeled + x2 weight) | **+.008** | +.019 | +.003 | .010 |
| Dyn. (50% relabeled + x3 weight) | +.005 | +.017 | **+.012** | **.011** |
| Dyn. (50% relabeled + x5 weight) | -.001 | **+.022** | +.008 | .009 |
| Dyn. (50% relabeled + x10 weight) | .000 | **+.022** | +.007 | .009 |

## 6.5 Summary and Conclusion

In this chapter, we have investigated how Concept Drift affects the accuracy of machine learning models applied to the detection of software vulnerability communication in social media and hacker forums. As the nature of computer security and language is dynamic, we confirm our expectation that the performance of these models might decrease with the rapidly changing concepts within a 1-year period. However, models trained and applied to Twitter messages do not show apparent drift in this same period. We believe that, compared to other hacker forums, Twitter provides shorter and less variable post content with regards to novelty on hacker techniques and exploitation tools. Thus, we imply that the drift on this platform is not enough to affect the performance of such models. Additionally, we imply that, as this social platform is used mostly by providing information to a large spectrum of people, the technical information posted by security researchers is provided in a more direct style, considering the non-expert audience. Thus, they do not deepen into the specificity of technical details and novel terms.

Furthermore, to provide a solution to the drift problem, we have investigated techniques such as periodic update and weighting. We have found that by updating the model with 50% of incoming instances (2 months period) while giving them weights in a proportion of 3:1 is sufficient to avoid the performance decreasing.

We acknowledge that, in a real-life situation, it still infeasible to perform the labelling of 50% of messages in larger streams of data. However, with these results, we highlight two important findings: (1) the importance of retraining the model over new (recent) messages and weight them to maintain the performance, and (2), not all streamed messages have the same importance, as the drift can be avoided by at least 50% of the relabelled data. We also point out that our experiment is limited to the amount of data we collected. It is recommended that further tests should be carried out using greater volumes of posts over longer periods.

CHAPTER 7

# Conclusion

This thesis has acknowledged that that software is in a paramount position for modern society. Today, people are more reliant on software-based devices due to their presence in every aspect of our lives. While software has been providing advances in communication technology, it also represents a big challenge in terms of security, privacy, availability, and reputation of software companies. Regarding software vulnerability, our recent software history has shown that these flaws are introduced (intentionally or not) and others will try to take advantage of this situation, for personal or financial reasons.

In this context, this research is leveraging the recent advances in machine learning to minimise this problem by investigating the occurrence of vulnerabilities communications in social media, thus providing an automatic model for acquiring information regarding malicious activities that exploit the flaws found in software products. Current approaches have been tackling this problem by improving the quality of the product during the development life cycle phase, however, these methods do not account for an already available (released) software. Moreover, in a security point of view, we cannot guarantee that all software providers are following the best practices for ensuring the quality and security of their products.

As a result, in this thesis, we have investigated the applicability of supervised machine learning and text classification techniques for automatically detect malicious communication regarding software exploitation in social media and hacker forums. Thus, the contributions of this work are based on a Cyber Security Intelligence approach, aiming to highlight any potential vulnerability being discussed in social network channels. A summary of the contributions provided in this thesis is in the following section.

## 7.1 Summary of Contributions

The contributions of this thesis have considered the application of these models in real-life situation. We highlight that the techniques used to build the classification models are not limited to the sources used in this thesis. They can be used also in other types of sources, for instance, Reddit, StackOrverflow, Github and for a different classification tasks, such as detection of private data leakage, detection of coordinate DDOS attacks communication or detection of user account and credit card number sharing. The summary of the contributions are listed below:

- **A new labelled dataset was provided in the domain of software vulnerability communication**. In Chapter 2, we have mentioned the lack of gold standard datasets in the security domain, especially in software vulnerability communication subdomain. This problem affects the reproducibility of results and further comparison. To fill this gap, we provided labelled datasets containing messages regarding vulnerabilities in software from 5 different social media sources and collected from 3 cyber domains (Surface web, Deep Web and Dark Web). These messages contain mentions of hacking tools, Proof of Concepts (PoC) descriptions, tutorials and exploits. Additionally, the timestamp of these messages is also provided. In Chapter 3, we described how we performed the selection of messages related to our research purpose. Furthermore, we described how they were systematically labelled by multiple annotators to guarantee that the ambiguity of the posts and the subjectivity of the annotator did not affect the quality of labels. Additionally, instead of using only binary labels for vulnerability communication, such as *Yes* and *No*, we have introduced a third label called Undecided to capture the real-world scenario where messages are not very explicit in their meaning.

- **Oversampling techniques on imbalanced datasets enhanced the performance of traditional classification models aimed to detect software vulnerability communication**. In Chapter 4, we provided experiments using different models create with 3 different datasets (Hacker forum, Twitter and Marketplace). In all three datasets, the classification performance improved when randomly oversampled techniques were applied in the minority class instances. The improvement can achieve a performance increase of 5%, on average. Additionally, to provide a "good enough" oversampling, we followed the *first optimal* rule, where the sampling proportion of 300% to 450% could be applied for increasing the performance without overfitting the model.

- **Optimal techniques for efficiency of traditional classification models**. In Chapter 4, through empirical experimentation, we provided the optimal approaches for reducing the dimensionality of traditional classification models using features selection and features extraction techniques. Therefore, we observed that we can reduce the number of features to less than 50% by removing the least frequent words (i.e., appears in less than 0.1% of messages) and most frequent words (i.e., appears in more than 20% of messages) without losing the overall accuracy of the model. On top of frequency reduction, we can reduce by a further 50% by applying the Chi-square Feature Selection or 90% by applying the SVD Feature Extraction.

- **Specific-vocabulary language models are potentially better than general-vocabulary language models for detection of software vulnerability communication**. In Chapter 5, we compared the use of specific-vocabulary and general-vocabulary language models in downstream task classification. We used the description of software vulnerabilities in NVD database to create a *word2vec* specific-vocabulary model. Despite the specific-vocabulary model being trained in much less data (3M tokens) compared to general-vocabulary model (100B tokens), it achieved a slightly better result in 2 of 3 datasets in both metrics, Positive Recall and Avg. Class Accuracy. This result indicates that, if the specific-vocabulary model was trained in more data, the resulting accuracy would improve

- **Sentence Embedding language models as features representation enhanced classification performance of models aimed to detect software vulnerability communication**. In Chapter 5, we provided a downstream classification task using a range of SEMB models as feature representation. As a result, SEMB as features representation is better suited for classifiers compared to WEMB. Additionally, models built with *SentEncoder* architecture (as features representation) has provided the best result in 5 of 5 datasets, compared to the other that uses SVM algorithm, achieving 74% of Positive Recall, followed by *word n-grams* and *char n-grams* with 55%, and WEMB, 33% (on average).

- **CNN and RNN-based Deep Learning architectures are better suited for detection software vulnerability communication compared to traditional algorithms**. In Chapter 5, we provided a downstream classification task using two different types of Deep learning architecture, the text CNN and BiLSTM. Both have achieved the best overall results. BiLSTM recorded, on average, 84% and 77%, of Avg. Class Accuracy and Positive Recall respectively, while text CNN recorded 96% and 93% in the same metrics.

- **Performance of models created with hacker forums content degrades in a less than 1-year period, indicating Concept Drift**. In Chapter 6, we simulated the real-life use of these classification models to evaluate the performance over a 1-year period. We performed a stream-based evaluation to identify whether the performance degrades. The performance recorded for models trained in hacker forums showed a tendency to decrease, indicating that they are more sensitive to Concept Drift within a 1-year period. However, for the model trained with Twitter dataset, the performance did not show the same behaviour. These results suggest that the models created with technical-vocabulary source (i.e., a source that contains a large number of specialised words as Hacker Forums), are more sensitive to changes in concept, thus degrade faster than models trained with general-purposed-vocabulary source (i.e., source aimed to communicate to a broad audience as Twitter).

- **Provided measures for avoiding the performance decrease of models over time, under Concept Drift effects**. In Chapter 6, we discovered that updating the model periodically is a simple method for avoiding the decrease of performance over time. Furthermore, by adding weights on the newest instances, we were able to enhance the performance of the models even more. Also, with the results, we could infer that there is no need to update the model with all incoming instances, as it has shown that updating with 50% of incoming instances provides comparable results to updating with 100%, which also help to reduce the workload needed to perform the labelling task. Finally, our results also suggest that the newest instances are very important for the performance of the model, as it has been shown that by adding weights only in the newest instances improves the final performance of the model.

## 7.2 Limitations

The first limitation of this work is in regards to the size of the datasets. Due to the limited amount of expert knowledge to perform the labelling task, we reduced the labelling task to 10,000 messages in total. This way, we could perform the labelling task 3 times (by different people), totalling 30,000 labelling actions, thus reducing possible bias and mistakes in the final label. We believe that, with more labelled examples, the performance of models could have been maintained for longer periods, avoiding early re-training, and reaching higher values, especially the traditional classification models.

The second point of limitation is in relation to the diversity of the dataset. With our collection of datasets, we have covered 3 Internet domains, Surface Web, Deep Web and Dark web. However, we have only touched a small part of the possible social media available on Internet, for instance, in Surface Web, we could have use StackOverflow, Reddit, or GitHub. Also, in Deep and Dark web, there are a large number of hacker forums and marketplaces being created at this moment that could be used as a source for this research. Thus, we believe that the results provided in this thesis could be reinforced by including more examples of these social media and forums.

Finally, the third limitation is related to the need to update the model with re-trained instances in order to maintain the performance of the model (mentioned in Chapter 6). As we have previously discussed, this approach requires expert knowledge on an ongoing basis. Also, as we mentioned, 50% re-labeled instances are suffice to avoid the decrease of model's performance. However, we acknowledge that, in a real-life situations, 50% might not help to reduce the labelling workload, as the volume of incoming messages might be large. We believe this limitation is an open research path with regards to finding what are the *worth-to-label* messages, in other words, the messages which are more likely to help the model to maintain its accuracy and reduce even more the workload needed for labelling those instances.

## 7.3 Future Work

Additionally, with regards to the suggestions mentioned in the limitation section, we propose the following path as promising future work:

Firstly, we have mentioned that the promotion of good labels are the bottleneck for efficient classification models. To guarantee the quality of these labels, expert knowledge is required. Such work might therefore be time- and resource-expensive depending on the volume of the dataset. As a result, we believe that further research could be done by sharing the labelled dataset among stakeholders (companies and research institutions), thus, reducing the workload needed to perform the task. However, if data needs to comply with privacy laws, the research might be extended to the field of *Homomorphic Encryption* applied to machine learning, where the main purpose is making use of encrypted information without necessarily decrypt it for training the models. This way, we could guarantee the confidentiality and privacy of eventual private information contained in the dataset. Furthermore, another approach could focus on a distributed system that allows the models being trained separately. However, the final weights (parameters) of the model can be shared and synchronised with other models. This approach has three benefits: (1) a distribution of computational cost and human workload, (2) the possibility of covering a large range of social media communities, (3) the improvement of model's performance.

Secondly, with regards to machine learning applied to text classification, we observed that this area has been evolving in a fast-pace, with state-of-the-art machine learning solutions surging very often. Since our experiments with Word and Sentence Embedding, new transform-based language models architecture has benchmarked in various NLP tasks, for instance, BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2018) and recently GPT-3 (Brown et al., 2020). These models have been improving due to the massive volume of data they were trained in and also their ability to fine-tune to a specific task (without the need of training in a specific-purpose corpus). However, training a state-of-the-art model, such as GPT-3, would cost at least $12M with GPU cloud services providers (INFO-Q, 2020). We believe the financial cost limits research in many institutions around the world but also opens new opportunities for researching feasible approaches to create a robust language model, in other words, studying forms of creating such models that use less computational resources.

Finally, the use of other NLP approaches, for instance, Named-entity Recognition (NER), could be applied for identifying fine-grained information on hacker messages. This way, we could extract the name of the software and company being hacked as well as the type of vulnerability, if mentioned in the sentence. Identifying such information is challenging due to the dynamic nature of hacker language, which is constantly including new terms, tools and techniques. The collected information could add more insights to further cyber security investigation. Furthermore, we believe that this research it is not only bound to the detection of software vulnerability communication but can also be applied to other malicious activities, for instance, detection of private data exposure and coordinate DDoS attacks.

# Bibliography

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938.

Ablon, L., Libicki, M. C., and Golay, A. A. (2014). *Markets for Cybercrime Tools and Stolen Data: Hackers' Bazaar*. RAND Corporation. URL: `http://www.jstor.org/stable/10.7249/j.ctt6wq7z6` [Accessed: Fev, 2020].

Acar, Y., Stransky, C., Wermke, D., Weir, C., Mazurek, M. L., and Fahl, S. (2017). Developers need support, too: A survey of security advice for software developers. In *2017 IEEE Cybersecurity Development (SecDev)*, pages 22–26.

Al Shalabi, L. and Shaaban, Z. (2006). Normalization as a preprocessing engine for data mining and the approach of preference matrix. In *2006 International Conference on Dependability of Computer Systems*, pages 207–214.

Algarni, A. and Malaiya, Y. (2014). Software vulnerability markets: Discoverers and buyers. *International Journal of Computer, Information Science and Engineering*, 8(3):71–81.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., and Kochut, K. (2017). A brief survey of text mining: Classification, clustering and extraction techniques. *preprint arXiv: 1707.02919*.

Amir, S., Wallace, B. C., Lyu, H., Carvalho, P., and Silva, M. J. (2016). Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of The*

*20th SIGNLL Conference on Computational Natural Language Learning*, pages 167–177, Berlin, Germany. Association for Computational Linguistics.

Bahassine, S., Madani, A., Al-Sarem, M., and Kissi, M. (2020). Feature selection using an improved chi-square for arabic text classification. *Journal of King Saud University - Computer and Information Sciences*, 32(2):225 – 231.

Bellman, R., Corporation, R., and Collection, K. M. R. (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

Benjamin, V. and Chen, H. (2012). Securing cyberspace: Identifying key actors in hacker communities. In *2012 IEEE International Conference on Intelligence and Security Informatics*, pages 24–29.

Benjamin, V., Li, W., Holt, T., and Chen, H. (2015). Exploring threats and vulnerabilities in hacker web: Forums, irc and carding shops. In *2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 85–90.

Benjamin, V. A. and Chen, H. (2013). Machine learning for attack vector identification in malicious source code. In *2013 IEEE International Conference on Intelligence and Security Informatics*, pages 21–23.

Benjamin, V. A. and Chen, H. (2015). Developing understanding of hacker language through the use of lexical semantics. *2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 79–84.

Bifet, A. and Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, pages 249–260, Berlin, Heidelberg. Springer-Verlag.

Bilge, L. and Dumitraş, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 833–844, New York, NY, USA. ACM.

Bleeping Computer (2020). Over 500,000 Zoom accounts sold on hacker forums, the dark web. URL: `https://www.bleepingcomputer.com/news/security/over-500-000-zoom-accounts-sold-on-hacker-forums-the-dark-web/` [Accessed: May, 2020].

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *preprint arXiv: 2005.14165.*

CBC.ca (2015). Heartbleed bug crisis hit more government computers than previously disclosed. URL: `https://www.cbc.ca/news/politics/heartbleed-bug-cyberattack-canada-government-departments-1.3355993` [Accessed: May, 2020].

Cer, D., Yang, Y., yi Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder. *preprint arXiv: 1803.11175.*

Chatterjee, S. and Thekdi, S. (2020). An iterative learning and inference approach to managing dynamic cyber vulnerabilities of complex systems. volume 193. URL: `https://tinyurl.com/y543wr63` [Accessed: Jun, 2020].

Chen, H., Mckeever, S., and Delany, S. J. (2017a). Harnessing the power of text mining for the detection of abusive content in social media. In *Advances in Computational Intelligence Systems*, pages 187–205, Cham. Springer International Publishing.

Chen, H., Mckeever, S., and Delany, S. J. (2017b). Presenting a labelled dataset for real-time detection of abusive user posts. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 884–890, New York, NY, USA. Association for Computing Machinery.

Chen, H., Mckeever, S., and Delany, S. J. (2019). The use of deep learning distributed representations in the identification of abusive text. In *13th International AAAI Conference on Web and Social Media ICWSM-2019*, volume 13, pages 125–133, Munich, Germany.

Chen, Q. and Bridges, R. A. (2017). Automated behavioral analysis of malware: A case study of wannacry ransomware. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 454–460.

Cherqi, O., Mezzour, G., Ghogho, M., and El Koutbi, M. (2018). Analysis of hacking related trade in the darkweb. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 79–84.

Chmielnicki, W. and Stąpor, K. (2016). Using the one–versus–rest strategy with samples balancing to improve pairwise coupling classification. *International Journal of Applied Mathematics and Computer Science*, 26(1):191 – 201.

CNBC (2019). The great Equifax mystery: 17 months later, the stolen data has never been found, and experts are starting to suspect a spy scheme. URL: `https://www.cnbc.com/2019/02/13/equifax-mystery-where-is-the-data.html` [Accessed: Jul, 2019].

CNET (2017). Buyers, sellers and cops on the hunt for AlphaBay's successor. URL: `https://www.cnet.com/news/`

alphabay-hansa-silk-road-dream-market-dark-web-shuts-down/ [Accessed: Jul, 2018].

CNET (2020a). Zoom passes 300M daily users as new security vulnerability discovered. URL: https://siliconangle.com/2020/04/23/zoom-numbers-pass-300m-daily-new-security-vulnerability-discovered/ [Accessed: May, 2020].

CNET (2020b). Zoom security issues: Zoom buys security company, aims for end-to-end encryption. URL: https://tinyurl.com/y794kmv3 [Accessed: May, 2020].

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning Journal*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Deepanshu Jindal (2019). Fine-grained analysis of sentence embeddings. URL: https://towardsdatascience.com/fine-grained-analysis-of-sentence-embeddings-a3ff0a42cce5 [Accessed: Dec, 2019].

Deibert, R., Rohozinski, R., Manchanda, A., Villeneuve, N., and Walton, G. (2009). Tracking ghostnet: investigating a cyber espionage network. Technical report. URL: http://www.nartv.org/mirror/ghostnet.pdf [Accessed: Jan, 2019].

Deliu, I., Leichter, C., and Franke, K. (2017). Extracting cyber threat intelligence from hacker forums: Support vector machines versus convolutional neural networks. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 3648–3656.

Deliu, I., Leichter, C., and Franke, K. (2018). Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5008–5013. IEEE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *preprint arXiv: 1810.04805*.

Do, Q., Martini, B., and Choo, K. R. (2016). A data exfiltration and remote exploitation attack on consumer 3d printers. *IEEE Transactions on Information Forensics and Security*, 11(10):2174–2186.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130.

Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 148–155, New York, NY, USA. Association for Computing Machinery.

Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M., and Halderman, J. A. (2014). The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 475–488, New York, NY, USA. Association for Computing Machinery.

D'Orazio, C. J., Choo, K. R., and Yang, L. T. (2017). Data exfiltration from internet of things devices: ios devices as case studies. *IEEE Internet of Things Journal*, 4(2):524–535.

Facebook Newsroom (2018). Security Update. URL: `https://newsroom.fb.com/news/2018/09/security-update/` [Accessed: Jul, 2019].

Feng, W., Sun, J., Zhang, L., Cao, C., and Yang, Q. (2016). A support vector machine based naive bayes algorithm for spam filtering. In *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8.

Fogel, D., Hanson, J. C., Kick, R., Malki, H. A., Sigwart, C., Stinson, M., Turban, E., and Rubin, S. H. (1993). The impact of machine learning on expert systems. In *Proceedings of the 1993 ACM Conference on Computer Science*, CSC '93, pages 522–527, New York, NY, USA. Association for Computing Machinery.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Machine Learning Research*, 3:1289–1305.

Fu, T., Abbasi, A., and Chen, H. (2010). A focused crawler for dark web forums. *Journal of the American Society for Information Science and Technology*, 61(6):1213–1231.

Gan, Z., Pu, Y., Henao, R., Li, C., He, X., and Carin, L. (2016). Learning generic sentence representations using convolutional neural networks. *preprint arXiv: 1611.07897*.

Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):1–6.

Ghafur, S., Kristensen, S., Honeyford, K., Martin, G., Darzi, A., and Aylin, P. (2019). A retrospective impact analysis of the wannacry cyberattack on the nhs. *npj Digital Medicine*, 2(1):98.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, volume 4, pages 2047–2052.

Greengard, S. (2016). Cybersecurity gets smart. *Communications of the ACM*, 59(5):29–31.

Grisham, J., Samtani, S., Patton, M., and Chen, H. (2017). Identifying mobile malware and key threat actors in online hacker forums for proactive cyber threat intelligence. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 13–18.

Guerra, P. C., Meira, W., and Cardie, C. (2014). Sentiment analysis on evolving social streams: How self-report imbalances can help. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 443–452, New York, NY, USA. Association for Computing Machinery.

Harrag, F. and El-Qawasmah, E. (2009). Neural network for arabic text classification. In *2009 Second International Conference on the Applications of Digital Information and Web Technologies*, pages 778–783.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *preprint arXiv: 1512.03385*.

Hidalgo, J. I. G., Maciel, B. I. F., and Barros, R. S. M. (2019). Experimenting with prequential variations for data stream learning evaluation. volume 35, pages 670–692.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hjortdal, M. (2011). China's use of cyber warfare: Espionage meets strategic deterrence. *Journal of Strategic Security*, 4(2):1–24.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.

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

Horawalavithana, S., Bhattacharjee, A., Liu, R., Choudhury, N., O. Hall, L., and Iamnitchi, A. (2019). Mentions of security vulnerabilities on reddit, twitter and github. In *IEEE/WIC/ACM International Conference on Web Intelligence*, WI '19, pages 200–207, New York, NY, USA. Association for Computing Machinery.

Huang, C., Liu, J., Fang, Y., and Zuo, Z. (2016). A study on web security incidents in china by analyzing vulnerability disclosure platforms. *Comput. Secur.*, 58(C):47–62.

Huang, Z., Dong, M., Mao, Q., and Zhan, Y. (2014). Speech emotion recognition using cnn. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 801–804, New York, NY, USA. Association for Computing Machinery.

Ikram, S. T. and Cherukuri, A. K. (2017). Intrusion detection model using fusion of chi-square feature selection and multi class svm. *Journal of King Saud University - Computer and Information Sciences*, 29(4):462 – 472.

INFO-Q (2020). OpenAI Announces GPT-3 AI Language Model with 175 Billion Parameters. URL: `https://www.cbc.ca/news/politics/heartbleed-bug-cyberattack-canada-government-departments-1.3355993` [Accessed: Jun, 2020].

Jang-Jaccard, J. and Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer and System Sciences*, 80(5):973 – 993.

Jianqiang, Z. and Xiaolin, G. (2017). Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5:2870–2879.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In Nédellec, C. and Rouveirol, C., editors, *Machine Learning: ECML-98*, pages 137–142, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jordaney, R., Sharad, K., Dash, S. K., Wang, Z., Papini, D., Nouretdinov, I., and Cavallaro, L. (2017). Transcend: Detecting concept drift in malware classification models. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 625–642, Vancouver, BC. USENIX Association.

Juniper Research (2017). Cybercrime and the internet of threats 2017. Technical report. URL: `https://www.juniperresearch.com/resources` [Accessed: Mar, 2019].

Kadam, S. V. (2019). A survey on classification of concept drift with stream data. URL: `https://hal.archives-ouvertes.fr/hal-02062610` [Accessed: Jan, 2020].

Kadhim, A. I., Cheah, Y., and Ahamed, N. H. (2014). Text document preprocessing and dimension reduction techniques for text document clustering. In *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, pages 69–73.

Kamath, C. N., Bukhari, S. S., and Dengel, A. (2018). *Comparative Study between Traditional Machine Learning and Deep Learning Approaches for Text Classification*. Association for Computing Machinery, New York, NY, USA.

Kameswara Sarma, P., Liang, Y., and Sethares, B. (2018). Domain adapted word embeddings for improved sentiment classification. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 51–59, Melbourne. Association for Computational Linguistics.

Kelleher, J. D., Namee, B. M., and D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press.

Khandpur, R. P., Ji, T., Jan, S., Wang, G., Lu, C.-T., and Ramakrishnan, N. (2017). Crowdsourcing cybersecurity: Cyber attack detection using social media. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1049–1057, New York, NY, USA. Association for Computing Machinery.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *preprint arXiv: 1412.6980*.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. *preprint arXiv: 1506.06726*.

Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y. (2019). Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19.

Kolter, J. Z. and Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.*, 7:2721–2744.

König, A. C. and Brill, E. (2006). Reducing the human overhead in text categorization. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 598–603, New York, NY, USA. Association for Computing Machinery.

krebsonsecurity.com (2020). 'War Dialing' Tool Exposes Zoom's Password Problems. URL: `https://krebsonsecurity.com/2020/04/war-dialing-tool-exposes-zooms-password-problems/` [Accessed: May, 2020].

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages 1188–1196. JMLR.org.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan-Kaufmann.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, H. and Kang, S. (2019). Word embedding method of sms messages for spam message filtering. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–4.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In Nédellec, C. and Rouveirol, C., editors, *Machine Learning: ECML-98*, pages 4–15, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lindsay, J. R. (2013). Stuxnet and the limits of cyber warfare. *Security Studies*, 22(3):365–404.

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Fogh, A., Horn, J., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., and Hamburg, M. (2018). Meltdown: Reading kernel memory from user space. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, pages 973–990, USA. USENIX Association.

Lippmann, R. P., Campbell, W. M., Weller-Fahy, D. J., Mensch, A. C., Zeno, G. M., and Campbell, J. P. (2016). Finding malicious cyber discussions in social media. *Lincoln Laboratory Journal*, 22(1):46–59.

Liu, H. (2017). Sentiment analysis of citations using word2vec. *preprint arXiv: 1704.00177*.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

LLC, P. I. (2017). Cost of cybercrime study. URL: `https://www.accenture.com/_acnmedia/PDF-62/Accenture-2017CostCybercrime-US-FINAL.pdf` [Accessed: Dec, 2018].

Macdonald, M., Frank, R., Mei, J., and Monk, B. (2015). Identifying digital threats in a hacker web forum. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 926–933, New York, NY, USA. Association for Computing Machinery.

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI workshop 1998*, pages 41–48.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *preprint arXiv: 1301.3781*.

Mittermayer, M. and Knolmayer, G. F. (2006). Newscats: A news categorization and trading system. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1002–1007.

Mulwad, V., Li, W., Joshi, A., Finin, T., and Viswanathan, K. (2011). Extracting information about security vulnerabilities from web text. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 257–260.

Nam, N. T. and Hung, P. D. (2019). Padding methods in convolutional sequence model: An application in japanese handwriting recognition. In *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, ICMLSC 2019, pages 138–142, New York, NY, USA. Association for Computing Machinery.

NIST (2019). National vulnerability database (nvd). URL: `https://nvd.nist.gov` [Accessed: Mar, 2020].

Nunes, E., Diab, A., Gunn, A., Marin, E., Mishra, V., Paliath, V., Robertson, J., Shakarian, J., Thart, A., and Shakarian, P. (2016). Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 7–12.

OWASP (2017). Top 10 Application Security Risks - 2017. URL: `https://www.owasp.org/index.php/Top_10-2017_Top_10` [Accessed: Jul, 2019].

Palacio-Niño, J.-O. and Berzal, F. (2019). Evaluation metrics for unsupervised learning algorithms. *preprint arXiv: 1905.05667*.

Palaz, D., Magimai.-Doss, M., and Collobert, R. (2015). Analysis of cnn-based speech recognition system using raw speech as input. In *Proceedings of Interspeech*, pages 11–15. ISCA.

Peng, F. and Schuurmans, D. (2003). Combining naive bayes and n-gram language models for text classification. In Sebastiani, F., editor, *Advances in Information Retrieval*, pages 335–350, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*, pages 1532–1543. Association for Computational Linguistics.

Perone, C. S., Silveira, R., and Paula, T. S. (2018). Evaluation of sentence embeddings in downstream and linguistic probing tasks. *preprint arXiv: 1806.06259*.

Portnoff, R. S., Afroz, S., Durrett, G., Kummerfeld, J. K., Berg-Kirkpatrick, T., McCoy, D., Levchenko, K., and Paxson, V. (2017). Tools for automated analysis of cybercriminal markets. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 657–666, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Pupillo, L., Ferreira, A., and Varisco, G. (2018). Software vulnerability disclosure in europe. *Centre for European Policy Studies (CEPS)*. URL: `https://tinyurl.com/y4vhc268` [Accessed: Jul, 2019].

Queiroz, A., Keegan, B., and Mtenzi, F. (2017). Predicting software vulnerability using security discussion in social media. In *16th European Conference on Information Warfare and Security, ECCWS*, pages 628–634, Dublin, Ireland.

Queiroz, A. L., Mckeever, S., and Keegan, B. (2019). Eavesdropping hackers: Detecting software vulnerability communication on social media using text mining. In *The Fourth International Conference on Cyber-Technologies and Cyber-Systems*, pages 41–48, Porto, Portugal.

Quinlan, J. R. (1986). Induction of decision trees. *MachineLearning*, 1(1):81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018). Language models are unsupervised multitask learners. URL: `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf` [Accessed: Fev, 2020].

Reimers, N. and Gurevych, I. (2017). Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *preprint arXiv: 1707.06799*.

Ren, Y., Zhang, Y., Zhang, M., and Ji, D. (2016). Context-sensitive twitter sentiment classification using neural network. In *AAAI*, pages 215–221.

Rescorla, E. (2005). Is finding security holes a good idea? *IEEE Security Privacy*, 3(1):14–19.

Rice, D. (2007). *Geekonomics: The Real Cost of Insecure Software*. Pearson Education, 1 edition.

Ross, G. J., Adams, N. M., Tasoulis, D. K., and Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recogn. Lett.*, 33(2):191–198.

Roumani, Y., Nwankpa, J. K., and Roumani, Y. F. (2016). Examining the relationship between firms financial records and security vulnerabilities. *Int. J. Information Management*, 36(6):987–994.

Roy, A., Park, Y., and Pan, S. (2017). Learning domain-specific word embeddings from sparse cybersecurity texts. *preprint arXiv: 1709.07470*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. (2014). Imagenet large scale visual recognition challenge. *preprint arXiv: 1409.0575*.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

Sabottke, C., Suciu, O., and Dumitras, T. (2015). Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 1041–1056, Washington, D.C. USENIX Association.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Process Management*, 24(5):513–523.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Samtani, S., Chinn, K., Larson, C., and Chen, H. (2016). Azsecure hacker assets portal: Cyber threat intelligence and malware analysis. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 19–24.

Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng (2006). Some effective techniques for naive bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466.

SANS (2019). Top 25 Most Dangerous Software Errors. URL: `https://www.sans.org/top25-software-errors` [Accessed: Jul, 2019].

Santos, M. S., Soares, J. P., Abreu, P. H., Araujo, H., and Santos, J. (2018). Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier]. *IEEE Computational Intelligence Magazine*, 13(4):59–76.

Saravanan, P., Clarke, S., Chau, D. H. P., and Zha, H. (2014). Latentgesture: Active user authentication through background touch analysis. In *Proceedings of the Second International Symposium of Chinese CHI*, Chinese CHI '14, pages 110–113, New York, NY, USA. Association for Computing Machinery.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Sethi, T. S. and Kantardzic, M. (2018). When good machine learning leads to bad security: Big data (ubiquity symposium). *Ubiquity*, 2018(May):1–14.

Sheng, V. S., Provost, F., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 614–622, New York, NY, USA. Association for Computing Machinery.

Shepherd, S. (2013). How do we define responsible disclosure? *Information Security Reading Room*. URL: `https://www.sans.org/reading-room/whitepapers/threats/paper/932` [Accessed: Jun, 2020].

Shirey, R. (2007). Internet security glossary, version 2. RFC 4949, RFC Editor. URL: `http://www.rfc-editor.org/rfc/rfc4949.txt` [Accessed: Mar, 2020].

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *preprint arXiv: 1409.1556*.

Steele, R. D. (1996). Open source intelligence: What is it? why is it important to the military? *American Intelligence Journal*, pages 35–41.

Suykens, J. A. K. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *preprint arXiv: 1409.4842*.

Tan, W., Hassanpour, S., Heagerty, P., Rundell, S., Suri, P., Huhdanpaa, H., James, K., Carrell, D., Langlotz, C., Organ, N., Meier, E., Sherman, K., Kallmes, D., Luetmer, P., Griffith, B., Nerenz, D., and Jarvik, J. (2018). Comparison of natural language processing rules-based and machine-learning systems to identify lumbar spine imaging findings related to low back pain. *Academic Radiology*, 25(11):1422–1432.

Tang, D., Qin, B., and Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432, Lisbon, Portugal. Association for Computational Linguistics.

theverge.com (2017). A new ransomware attack is infecting airlines, banks, and utilities across Europe. URL: `https://www.theverge.com/2017/6/27/15879480/petrwrap-virus-ukraine-ransomware-attack-europe-wannacry` [Accessed: May, 2020].

Thompson, B., Morris-King, J., and Cam, H. (2016). Controlling risk of data exfiltration in cyber networks due to stealthy propagating malware. In *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pages 479–484.

Tiftikci, M., Özgür, A., He, Y., and Hur, J. (2019). Machine learning-based identification and rule-based normalization of adverse drug reactions in drug labels. *BMC Bioinformatics*, 20. Publisher Copyright: © 2019 The Author(s). Copyright: Copyright 2020 Elsevier B.V., All rights reserved.

Trabelsi, S., Plate, H., Abida, A., Aoun, M. M. B., Zouaoui, A., Missaoui, C., Gharbi, S., and Ayari, A. (2015). Mining social networks for software vulnerabilities monitoring. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–7.

Urbanowicz, R. J. and Moore, J. H. (2015). Exstracs 2.0: description and evaluation of a scalable learning classifier system. *Evolutionary Intelligence*, 8(2):89–116.

Uysal, A. K. and Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing and Management*, 50(1):104 – 112.

van der Maaten, L., Postma, E. O., and van den Herik, H. J. (2008). Dimensionality reduction: A comparative review.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *preprint arXiv: 1706.03762*.

Wang, D., Feng, S., Wang, D., and Yu, G. (2013). Detecting opinion drift from chinese web comments based on sentiment distribution computing. In Lin, X., Manolopoulos, Y., Srivastava, D., and Huang, G., editors, *Web Information Systems Engineering – WISE 2013*, pages 72–81, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wang, W. and Zhou, Z.-H. (2015). Crowdsourcing label quality: a theoretical analysis. *Science China Information Sciences*, 58(11):1–12.

Wang, X., Liu, Y., Sun, C., Wang, B., and Wang, X. (2015). Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1343–1353, Beijing, China. Association for Computational Linguistics.

Welinder, P. and Perona, P. (2010). Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 25–32.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *preprint arXiv: 1511.08198*.

Won, M. and Lee, J. (2018). Embedding for out of vocabulary words considering contextual and morphosyntactic information. In *2018 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, pages 212–215.

Wu, Z., Zheng, X., and Dahlmeier, D. (2017). Character-based text classification using top down semantic model for sentence representation. *preprint arXiv: 1705.10586*.

Yang, X., Macdonald, C., and Ounis, I. (2018). Using word embeddings in twitter election classification. *Information Retrieval Journal*, 21(2):183–207.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 42–49, New York, NY, USA. Association for Computing Machinery.

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2017). Recent trends in deep learning based natural language processing. *preprint arXiv: 1708.02709*.

Younis, A., Malaiya, Y. K., and Ray, I. (2016). Assessing vulnerability exploitability risk using software properties. *Software Quality Journal*, 24(1):159–202.

Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, 31(7):1235–1270.

ZDnet (2018). This is how much the WannaCry ransomware attack cost the NHS. URL: `https://www.zdnet.com/article/this-is-how-much-the-wannacry-ransomware-attack-cost-the-nhs/` [Accessed: May, 2020].

Zhang, T. and Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31.

Zhang, X., Zhao, J. J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *preprint arXiv: 1509.01626*.

Zhang, Y. and Wallace, B. C. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *preprint arXiv: 1510.03820*.

Zhao, K., Zhang, Y., Xing, C., Li, W., and Chen, H. (2016). Chinese underground market jargon analysis based on unsupervised learning. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 97–102.

Zliobaite, I., Pechenizkiy, M., and Gama, J. (2016). *An Overview of Concept Drift Applications*, pages 91–114. Studies in Big Data. Springer International Publishing AG, Switzerland.

# APPENDIX A1

# Keywords

Here we see the keywords used to filter the messages of the original dataset.

## A1.1 OWASP top 10 Application Security Risk

Injection, Broken Authentication, Sensitive Data Exposure, XML External Entities XXE, Broken Access Control, Security Misconfiguration, Cross-Site Scripting XSS, Insecure Deserialization, Insufficient Logging Monitoring

## A1.2 SANS top 25 Software Errors

Improper Neutralization Special Elements used Command SQL Injection, Improper Neutralization Special Elements used Command OS Command Injection, Improper Neutralization Input Web Page Generation Cross-site Scripting, Unrestricted Upload File Dangerous Type, Cross-Site Request Forgery CSRF, URL Redirection Untrusted Site Open Redirect, Buffer Copy without Checking Size Input, Classic Buffer Overflow, Improper Limitation Pathname Restricted Directory, Path Traversal, Download Code Without Integrity Check, Inclusion Functionality Untrusted Control Sphere, Use

Potentially Dangerous Function, Incorrect Calculation Buffer Size, Uncontrolled Format String, Integer Overflow Wraparound, Missing Authentication Critical Function, Missing Authorization, Use Hard-coded Credentials, Missing Encryption Sensitive Data, Reliance Untrusted Inputs Security Decision, Execution Unnecessary Privileges, Incorrect Authorization, Incorrect Permission Assignment Critical Resource, Use Broken Risky Cryptographic Algorithm, Improper Restriction Excessive Authentication Attempts, Use One-Way Hash without Salt

# APPENDIX A2

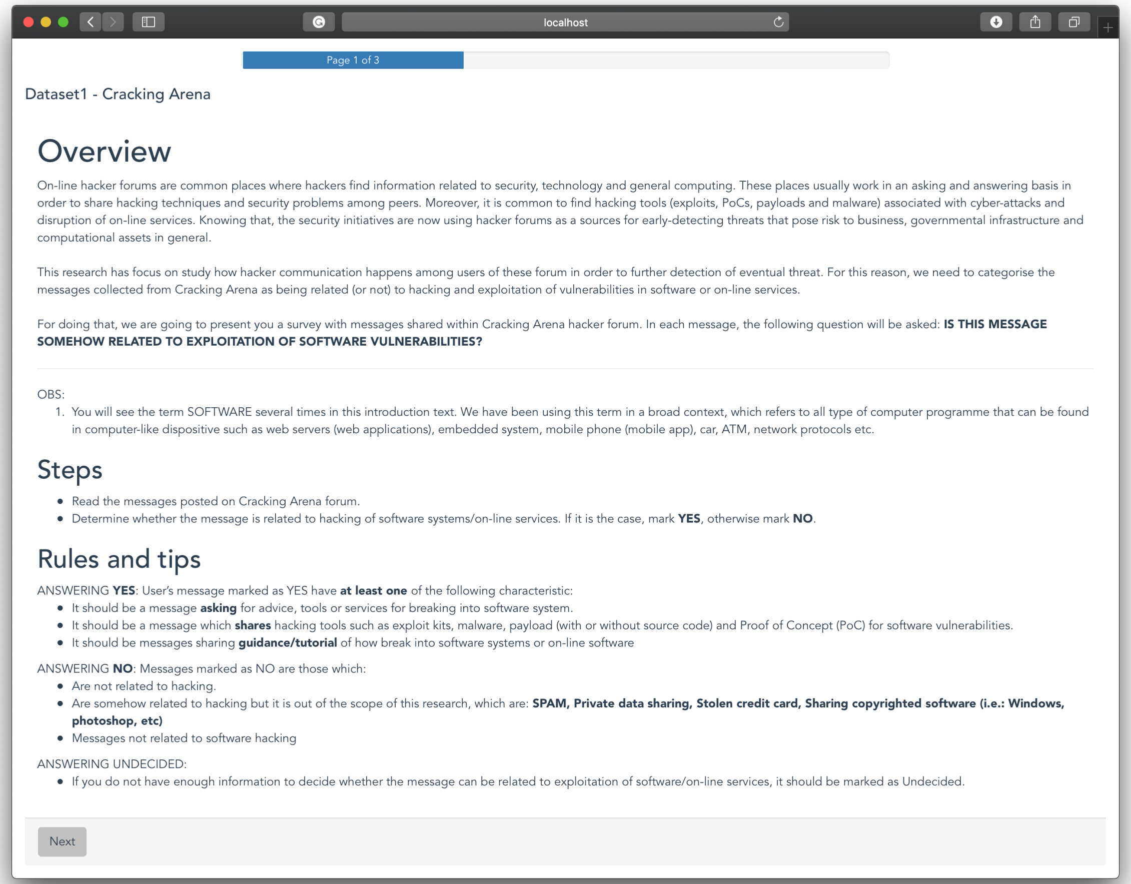
# Survey Pages and Introductory Text

## A2.1 D1 - Cracking Arena, D4 - Garage4hackers and D5 - Cracking Fire

On-line hacker forums are common places where hackers find information related to security, technology and general computing. These places usually work in an asking and answering basis in order to share hacking techniques and security problems among peers. Moreover, it is common to find hacking tools (exploits, PoCs, payloads and malware) associated with cyber-attacks and disruption of on-line services. Knowing that, the security initiatives are now using hacker forums as a sources for early-detecting threats that pose risk to business, governmental infrastructure and computational assets in general.

This research has focus on study how hacker communication happens among users of these forum in order to further detection of eventual threat. For this reason, we need to categorise the messages collected from Cracking Arena as being related (or not) to hacking and exploitation of vulnerabilities in software or on-line services.

Page 1 - Introduction Text - for All Hacker Forums



Dataset1 - Cracking Arena

## Overview

On-line hacker forums are common places where hackers find information related to security, technology and general computing. These places usually work in an asking and answering basis in order to share hacking techniques and security problems among peers. Moreover, it is common to find hacking tools (exploits, PoCs, payloads and malware) associated with cyber-attacks and disruption of on-line services. Knowing that, the security initiatives are now using hacker forums as a sources for early-detecting threats that pose risk to business, governmental infrastructure and computational assets in general.

This research has focus on study how hacker communication happens among users of these forum in order to further detection of eventual threat. For this reason, we need to categorise the messages collected from Cracking Arena as being related (or not) to hacking and exploitation of vulnerabilities in software or on-line services.

For doing that, we are going to present you a survey with messages shared within Cracking Arena hacker forum. In each message, the following question will be asked: **IS THIS MESSAGE SOMEHOW RELATED TO EXPLOITATION OF SOFTWARE VULNERABILITIES?**

OBS:

1. You will see the term SOFTWARE several times in this introduction text. We have been using this term in a broad context, which refers to all type of computer programme that can be found in computer-like dispositive such as web servers (web applications), embedded system, mobile phone (mobile app), car, ATM, network protocols etc.

## Steps

- Read the messages posted on Cracking Arena forum.
- Determine whether the message is related to hacking of software systems/on-line services. If it is the case, mark **YES**, otherwise mark **NO**.

## Rules and tips

ANSWERING **YES**: User's message marked as YES have **at least one** of the following characteristic:

- It should be a message **asking** for advice, tools or services for breaking into software system.
- It should be a message which **shares** hacking tools such as exploit kits, malware, payload (with or without source code) and Proof of Concept (PoC) for software vulnerabilities.
- It should be messages sharing **guidance/tutorial** of how break into software systems or on-line software

ANSWERING **NO**: Messages marked as NO are those which:

- Are not related to hacking.
- Are somehow related to hacking but it is out of the scope of this research, which are: **SPAM, Private data sharing, Stolen credit card, Sharing copyrighted software (i.e.: Windows, photoshop, etc)**
- Messages not related to software hacking

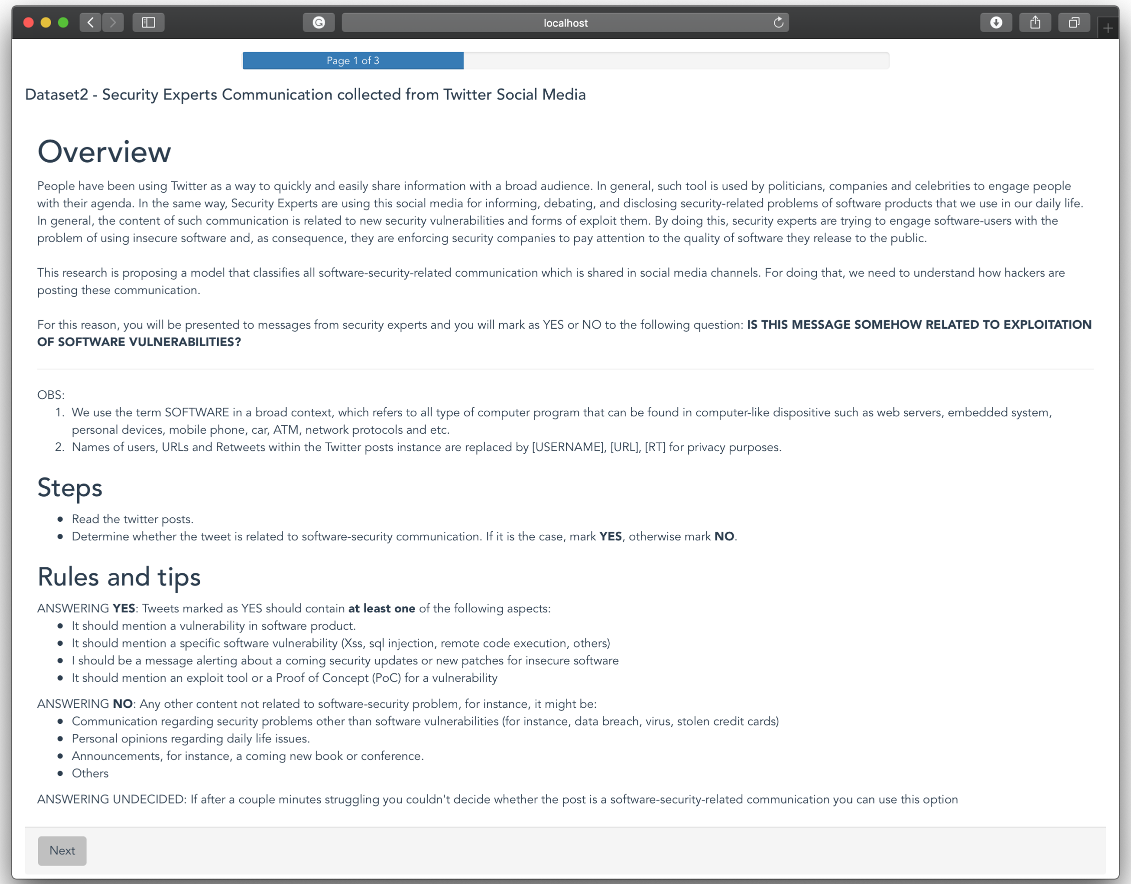
ANSWERING UNDECIDED:

- If you do not have enough information to decide whether the message can be related to exploitation of software/on-line services, it should be marked as Undecided.

## A2.2 D2 - Twitter Security Experts

People have been using Twitter as a way to quickly and easily share information with a broad audience. In general, such tool is used by politicians, companies and celebrities to engage people with their agenda. In the same way, Security Experts are using this social media for informing, debating, and disclosing security-related problems of software products that we use in our daily life. In general, the content of such communication is related to new security vulnerabilities and forms of exploit them. By doing this, security experts are trying to engage software-users with the problem of using insecure software and, as consequence, they are forcing security companies to pay attention to the quality of software they release to the public.

Page 1 - Introduction Text - Twitter Security Experts



Dataset2 - Security Experts Communication collected from Twitter Social Media

Overview

People have been using Twitter as a way to quickly and easily share information with a broad audience. In general, such tool is used by politicians, companies and celebrities to engage people with their agenda. In the same way, Security Experts are using this social media for informing, debating, and disclosing security-related problems of software products that we use in our daily life. In general, the content of such communication is related to new security vulnerabilities and forms of exploit them. By doing this, security experts are trying to engage software-users with the problem of using insecure software and, as consequence, they are enforcing security companies to pay attention to the quality of software they release to the public.

This research is proposing a model that classifies all software-security-related communication which is shared in social media channels. For doing that, we need to understand how hackers are posting these communication.

For this reason, you will be presented to messages from security experts and you will mark as YES or NO to the following question: **IS THIS MESSAGE SOMEHOW RELATED TO EXPLOITATION OF SOFTWARE VULNERABILITIES?**

OBS:

1. We use the term SOFTWARE in a broad context, which refers to all type of computer program that can be found in computer-like dispositive such as web servers, embedded system, personal devices, mobile phone, car, ATM, network protocols and etc.
2. Names of users, URLs and Retweets within the Twitter posts instance are replaced by [USERNAME], [URL], [RT] for privacy purposes.

Steps

- Read the twitter posts.
- Determine whether the tweet is related to software-security communication. If it is the case, mark **YES**, otherwise mark **NO**.

Rules and tips

ANSWERING **YES**: Tweets marked as YES should contain **at least one** of the following aspects:

- It should mention a vulnerability in software product.
- It should mention a specific software vulnerability (Xss, sql injection, remote code execution, others)
- I should be a message alerting about a coming security updates or new patches for insecure software
- It should mention an exploit tool or a Proof of Concept (PoC) for a vulnerability

ANSWERING **NO**: Any other content not related to software-security problem, for instance, it might be:

- Communication regarding security problems other than software vulnerabilities (for instance, data breach, virus, stolen credit cards)
- Personal opinions regarding daily life issues.
- Announcements, for instance, a coming new book or conference.
- Others

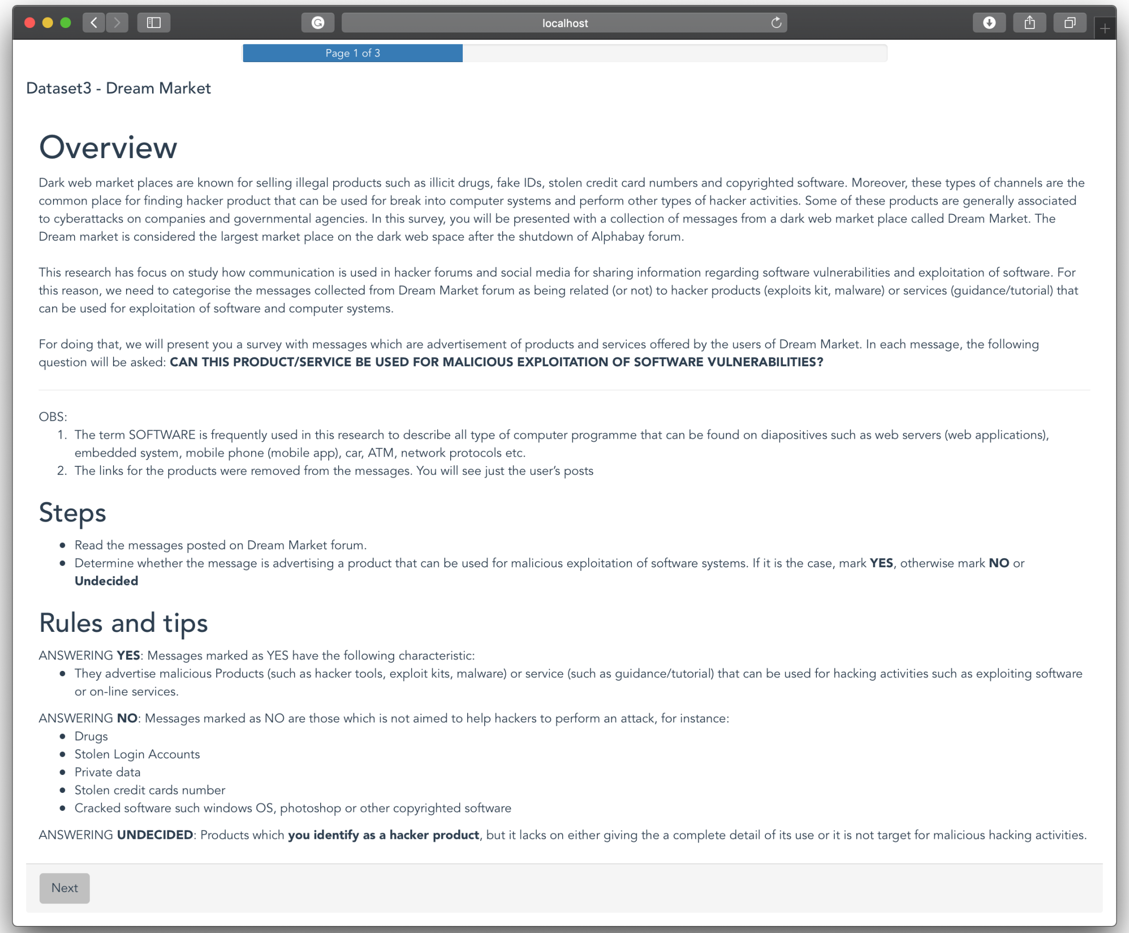
ANSWERING UNDECIDED: If after a couple minutes struggling you couldn't decide whether the post is a software-security-related communication you can use this option

### A2.3 D3 - Dream Market Marketplace

Dark web marketplaces are known for selling illegal products such as illicit drugs, fake IDs, stolen credit card numbers and copyrighted software. Moreover, these type of channels are a common place for finding hacker product that can be used for break into computer systems and perform other types of hacker activities. Some of these products are generally associated to cyber-attacks on companies and governmental agencies. In this survey, you will be presented with a collection of messages from a Dark Web marketplace called Dream Market. This forum is known as a place for buying and selling illicit products, among them, hacking tools and other resources for malicious activities. The Dream market is considered the largest market place on the dark web space after the shutdown of Alphabay forum.

This research has focus on study how communication is used in hacker forums and social media for sharing information regarding software vulnerabilities and exploitation of software. For this reason, we need to categorise the messages collected from Dream Market forum as being related (or not) to hacker products (exploits kit, malware) or services (guidance/tutorial) that can be used for exploitation of software and computer systems.

Page 1 - Introduction Text - Dream Market Marketplace



Dataset3 - Dream Market

## Overview

Dark web market places are known for selling illegal products such as illicit drugs, fake IDs, stolen credit card numbers and copyrighted software. Moreover, these types of channels are the common place for finding hacker product that can be used for break into computer systems and perform other types of hacker activities. Some of these products are generally associated to cyberattacks on companies and governmental agencies. In this survey, you will be presented with a collection of messages from a dark web market place called Dream Market. The Dream market is considered the largest market place on the dark web space after the shutdown of Alphabay forum.

This research has focus on study how communication is used in hacker forums and social media for sharing information regarding software vulnerabilities and exploitation of software. For this reason, we need to categorise the messages collected from Dream Market forum as being related (or not) to hacker products (exploits kit, malware) or services (guidance/tutorial) that can be used for exploitation of software and computer systems.

For doing that, we will present you a survey with messages which are advertisement of products and services offered by the users of Dream Market. In each message, the following question will be asked: **CAN THIS PRODUCT/SERVICE BE USED FOR MALICIOUS EXPLOITATION OF SOFTWARE VULNERABILITIES?**

OBS:
1. The term SOFTWARE is frequently used in this research to describe all type of computer programme that can be found on diapositives such as web servers (web applications), embedded system, mobile phone (mobile app), car, ATM, network protocols etc.
2. The links for the products were removed from the messages. You will see just the user's posts

## Steps

- Read the messages posted on Dream Market forum.
- Determine whether the message is advertising a product that can be used for malicious exploitation of software systems. If it is the case, mark **YES**, otherwise mark **NO** or **Undecided**

## Rules and tips

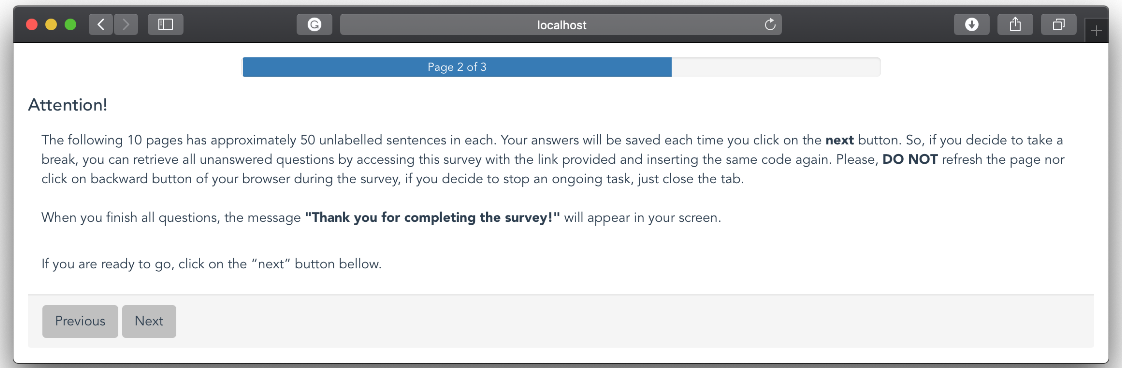
ANSWERING **YES**: Messages marked as YES have the following characteristic:
- They advertise malicious Products (such as hacker tools, exploit kits, malware) or service (such as guidance/tutorial) that can be used for hacking activities such as exploiting software or on-line services.

ANSWERING **NO**: Messages marked as NO are those which is not aimed to help hackers to perform an attack, for instance:
- Drugs
- Stolen Login Accounts
- Private data
- Stolen credit cards number
- Cracked software such windows OS, photoshop or other copyrighted software

ANSWERING **UNDECIDED**: Products which **you identify as a hacker product**, but it lacks on either giving the a complete detail of its use or it is not target for malicious hacking activities.

## A2.4 Common pages

Page 2 - Rules for Completing the Survey



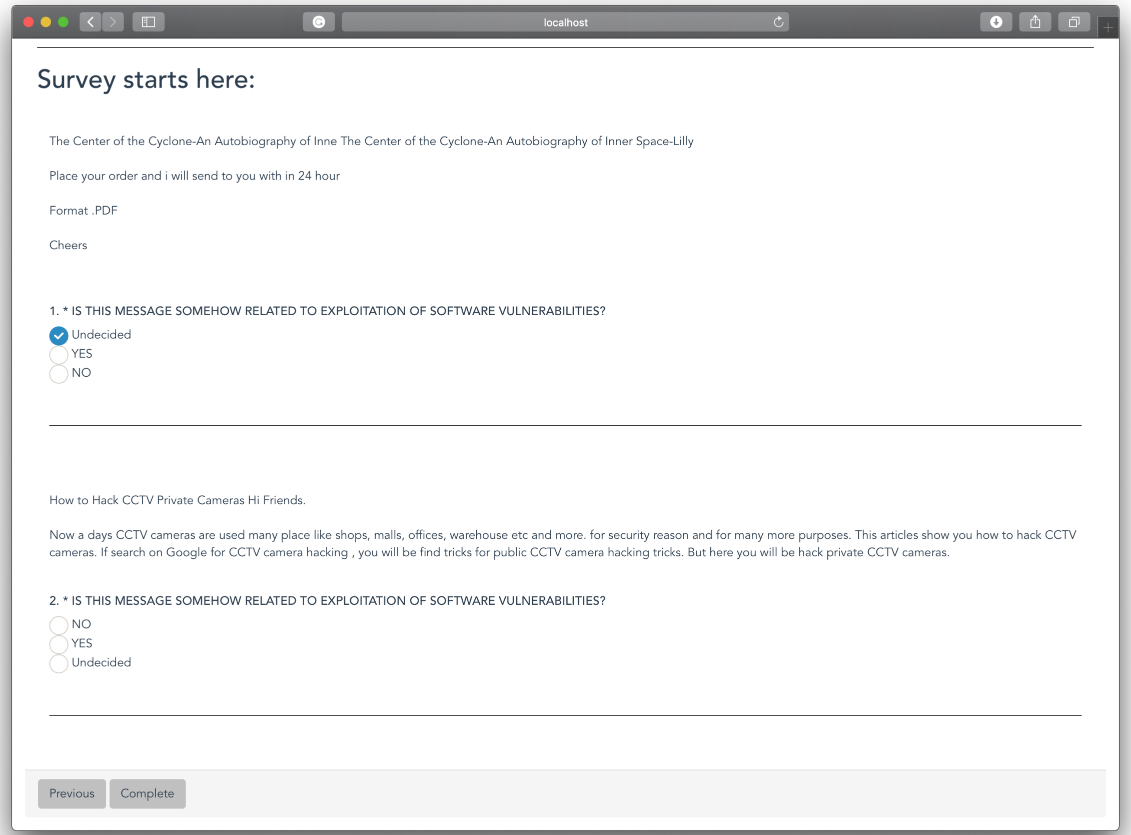
Attention!

The following 10 pages has approximately 50 unlabelled sentences in each. Your answers will be saved each time you click on the **next** button. So, if you decide to take a break, you can retrieve all unanswered questions by accessing this survey with the link provided and inserting the same code again. Please, **DO NOT** refresh the page nor click on backward button of your browser during the survey, if you decide to stop an ongoing task, just close the tab.

When you finish all questions, the message **"Thank you for completing the survey!"** will appear in your screen.

If you are ready to go, click on the "next" button bellow.

Page 3 - Survey



Survey starts here:

The Center of the Cyclone-An Autobiography of Inne The Center of the Cyclone-An Autobiography of Inner Space-Lilly

Place your order and i will send to you with in 24 hour

Format .PDF

Cheers

1. * IS THIS MESSAGE SOMEHOW RELATED TO EXPLOITATION OF SOFTWARE VULNERABILITIES?

[✓ Undecided]
YES
NO

How to Hack CCTV Private Cameras Hi Friends.

Now a days CCTV cameras are used many place like shops, malls, offices, warehouse etc and more. for security reason and for many more purposes. This articles show you how to hack CCTV cameras. If search on Google for CCTV camera hacking , you will be find tricks for public CCTV camera hacking tricks. But here you will be hack private CCTV cameras.

2. * IS THIS MESSAGE SOMEHOW RELATED TO EXPLOITATION OF SOFTWARE VULNERABILITIES?

NO
YES
Undecided