IMPLEMENTING NATURAL LANGUAGE PROCESSING TECHNIQUES FOR THE DETECTION OF STEGOSPAM FILES GENERATED USING A PROBABILISTIC CONTEXT-FREE GRAMMAR

by

FARAD HOOSEN CHOHAN

submitted in accordance with the requirements for
the degree of

MASTER OF TECHNOLOGY

in the subject

INFORMATION TECHNOLOGY

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: DR WYNAND VAN STADEN

(15 January 2021)

DECLARATION

Name: _____**Farad Hoosen Chohan**_____

Student number: ____**3134 -673-1**_____

Degree:_____**Master of Technology: Information Technology**

Exact wording of the title of the dissertation as appearing on the electronic copy submitted for examination:

IMPLEMENTING NATURAL LANGUAGE PROCESSING TECHNIQUES FOR THE DETECTION OF STEGOSPAM FILES GENERATED USING A PROBABILISTIC CONTEXT-FREE GRAMMAR

I declare that the above dissertation is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

I further declare that I submitted the dissertation to originality checking software and that it falls within the accepted requirements for originality.

I further declare that I have not previously submitted this work, or part of it, for examination at UNISA for another qualification or at any other higher education institution.

(The dissertation will not be examined unless this statement has been submitted.)

_____
SIGNATURE

11 January 2021
_____
DATE

Acknowledgements

Firstly, I would like to thank God, for affording me the opportunity, knowledge and conviction to persevere throughout this journey.

I would like to thank my supervisor, Dr. Wynand van Staden, for all his support, advice and guidance.

I would like to thank my baby girl Ariyaah, who at such a tender age understood and accepted the sacrifices that needed to be made to permit me to complete this paper.

Lastly, I would like to thank my awesome wife Amara, for sacrificing so much of her time and for carrying the family torch to allow me the time and solitude to complete this paper. She was as important to this paper getting done as I was. Thank you, Bella.

Abstract

The ubiquitous and innocuous nature of spam email makes it an ideal carrier for covert-based communications, particularly linguistic steganography, which seeks to conceal sensitive information within a body of text through the use of lexical encoding schemes. Stegospam generated using a Probabilistic Content-Free Grammar (PCFG) such as Spammimic which employs linguistic steganography techniques is extremely difficult to detect. Existing steganalysis approaches employ a combination of techniques that are either too specific to a particular stego-text or tool, or it requires the original text to perform a comparative analysis to discover lexical irregularities that may indicate the existence of a concealed message. Advancements in Natural Language Processing (NLP) has allowed for its application across various fields including cyber-security. Hence, this research evaluates the implementation of NLP techniques for the detection or classification of stegospam generated using Spammimic.

An experimental approach is adopted to evaluate the ability an NLP-based steganalysis to detect stegospam from non-stegospam files contained within known corpora. This is achieved through the implementation of software prototype constructed on NLP algorithms and methods.

The results of the research demonstrates the ability of the proposed steganalysis to successfully classify non-stegospam from stegospam files generated using Spammimic.

Keywords: Classification; classification algorithm; covert communications; feature extraction; linguistic steganalysis; lemmatization; Naïve Bayes classifier; natural language processing; sentence disambiguation; Spammimic; steganography; steganographic medium; stegospam; support vector machine; tokenization

Contents

Abbreviations and Acronyms

ASCII	American Standard Code for Information Interchange
ANN	Artificial Neural Network
B2B	Business to Business
B2C	Business to Customer
BEC	Business Email Compromise
C&C	Command & Control (C&C) Servers
CBOW	Continuous-Bag-Of-Words
CFG	Context-Free Grammars
CFPG	Context-Free Probabilistic Grammar
CRF	Conditional Random Field
DKIM	DomainKeys Identified Mail
DNS	Domain Name Server
DOS	Denial of Service
DDOS	*Distributed Denial of Service*
FTP	File Transfer Protocol
GNB	Gaussian Naïve Bayes
IaaS	Infrastructure as a Service
IDF	Inverse Document Frequency
IDS	Intrusion Detection System
IMAP	Internet Mail Transfer Protocol (Server)
IMAP4	Internet Message Access Protocol version4
IP	Internet Protocol
IS	Information Systems
ISP	Internet Service Providers
LSA	Latent Semantic Analysis
LSB	Least Significant Bit
MDA	Mail Delivery Agent
MIME	Multi-Purpose Internet Mail Extension
ML	Machine Learning (process)

MTA	Message Transfer Agent
MUA	Mail User Agent
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NLTK	Natural Language Tool Kit
NLU	Natural Language Understanding
NNLM	Neural Network Language Model
OCR	Optical Character Recognition (software)
PCFG	Probabilistic Context Free Grammar
PGP	Pretty Good Privacy
POP	Post office Protocol (Server)
POS	part-of-speech
RDF	Random Decision Forest
RFP	Relative Frequency Probability
ROI	Return on Investment
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework
SPF.txt	Sender Policy Framework (text)
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UBE	Unsolicited Bulk Email
UCE	Unsolicited Commercial Email
WLAN	Wireless Local Area Network
WSD	Word Sense Disambiguation

List of Figures

List of Tables

1

Research Questions, Objectives and Hypothesis

This chapter presents the research questions, objectives and hypothesis. It also provides a discussion on the significance and limitations of this research study. Lastly, an overview of the chapters is presented.

Introduction

Over the past few decades, the proliferation and significance of text-based data has increased exponentially, stimulated by the growth of the Internet and the significant role that it plays in everyday life in terms of the distribution of text media such as blogs, email and text messaging. The upsurge of text media directly raises concerns regarding its usage as a channel for covert-based communications (Castiglione, et al., 2012). Such covert means of communication is termed 'steganography'. Steganographic methods aim to conceal a secret message in a cover medium in a covert manner such that the presence of the embedded message in the resultant stego-object is imperceptible to anyone except the intended recipient (Huanhuan, et al., 2017).

It is practically impossible to think about email without considering the issue of spam. In March 2020, spam email accounted for 53.95% of the 306 billion emails sent and received worldwide, which include billions of promotional e-mails (Radicatic, 2020; Clement, 2020). Significantly, however, whilst many e-mail users consider email spam as innocuous and that such content belongs in their spam folder, not all spam are benign promotional e-mails. A significant portion are of a more malicious nature that aim to hijack user systems or are used in more sophisticated attacks known as 'spear-phishing' that are directed at individuals or organisations to elicit sensitive information (Clement, 2020; Satterfield, 2006).

This research considers spam email as the preferred cover medium for covert communications through the implementation of linguistic steganography used to create stegospam files. Additionally, this research study proposes a steganalysis approach implementing Natural Language Processing (NLP) methods to classify stego from non-stegospam files. The results recorded from this research study are analysed and conclusions are presented thereof.

Structure of this chapter:

Section 1.1 Provides a background of the research

Section 1.2 Presents the problem statement of this research

Section 1.3 Discusses the primary and secondary research questions

Section 1.4 Discusses the research objectives

Section 1.5 The Hypothesis is presented

Section 1.6 Discusses the delineations and limitations of this research

Section 1.7 Elaborates on the significance of this research

Section 1.8 Provides an overview for the rest of the chapters of this research

1.1 Background

Steganography, the art of transmitting information covertly through an open channel, is realized by concealing classified information within innocuous cover-objects such as digital images, text, audio or videos rendering it imperceptible to anyone except the intended recipient (Chang & Clark, 2014). The origin of steganography dates back to ancient Rome and is derived from the Greek terms – 'stego' which means concealment and 'graphy' which means art, study or style of writing (Chang and Clark, 2014; Trovi, ShivarKumar and Das, 2016). Modern day application of steganography comprises of digital watermarking, digital signatures, prevention of data manipulation and network traffic analysis. Additionally, the most common steganographic modality is image-based steganography - where a text payload in concealed within a digital image file (Trovi, ShivarKumar and Das, 2016).

The rationale behind steganography is the imperceptibility of a secret message concealed within an innocuous cover medium, as opposed to cryptography where the intention is not to hide the existence of the secret message but to make it indecipherable (Castiglione, et al., 2012; Changder and Majumder, 2013). Consequently, when deciding on the cover medium steganographers have to consider three critical factors: payload capacity, imperceptibility and robustness (Ahvanooey, et al., 2019). Owing to their file sizes and nature, multimedia files have a much higher payload capacity than plaintext files. However, innovative advancements in digital scanners and steganalysis tools have targeted multimedia files hence, detecting the 'noise' in the digital media has become a lot easier and scrutinising it against a pattern could result in the detection of a secret message. On the contrary, text-based steganography is a lot more robust and is not susceptible this form of detection (Trovi, ShivarKumar and Das, 2016). Attributed to this factor and the ubiquitous nature of digital text this research study considers text-based steganography in particular linguistic steganography.

Linguistic steganography is a form of steganography that employs innovative techniques to conceal sensitive information within a natural language text by means of lexical (semantic), statistical and syntactic transformation of the cover-text (Chang and Clark, 2014). Consequently, linguistic steganography is one of the more difficult forms of steganography to implement, as text files offer very little redundancy to conceal larger payloads in contrast to other forms of steganography (Chang and Clark, 2014; Sharma, Rekha and Manisha, 2016).

One of the most pervasive sources of natural language text is found in spam email messages, and due to its presumed innocuity, it serves as a quintessential carrier for covert communications (Castiglione, et al., 2012). To evade any suspicion from a third party observing the communication channel, stegospam files should not deviate too much from the 'traditional' appearance of regular spam files. A stegospam message or file is a cover-object that mimics the properties and appearance of an innocuous spam message. Additionally, steganographers can auto-generate the text of stegospam messages using a probabilistic context free grammar (PCFG) such as Spammimic, which implements an NLP technique known as natural language generation (NLG). Natural language generation is discussed in greater detail in Chapter 4 – Natural Language Processing.

Detecting stego-text files generated using a PCFG remains one of the most challenging tasks in the study of linguistic steganalysis (i.e. the process of detecting whether there are secret messages in digital text files). Most linguistic steganalysis approaches require the original cover text to compare and identify any statistical or linguistical distortions in the stego-text as demonstrated by Zhi-li, et al. (2008), and by Xiang, et al. (2018). Additionally, there has been very little work done on proposing an effective linguistic steganalysis for the detection of stegospam files generated using a PCFG. Consequently, this research study will consider and measure the implementation of various Natural Language Processing (NLP) methods or algorithms to effectively detect or classify stegospam from non-stegospam files.

1.2 Problem Statement

Over the past few decades, email has remained an open medium – i.e. if you know someone's email address you can send an email to them. It offers seamless interoperability among users across a wide range of heterogeneous devices (Clark, et al., 2018). As of March 2020, there were 4 billion email users worldwide sending more than 306 billion emails daily (Radicatic, 2020). Despite its ubiquity, email was developed without security in mind and remains largely insecure (Clark, et al., 2018). Additionally, email systems are highly susceptible to spam and other forms of cyber-attacks. Despite the volume of work done to filter spam emails, a fair percentage of spam emails does find its way into a user's inbox, where it is considered an

annoyance rather than a security threat (Bujang and Hussin, 2013). It is the sheer ubiquity and assumed innocuity of spam email that presents attackers with the opportunity to use spam email as a medium for covert communications (Castiglione, et al., 2012).

Consequently, one such approach which is the focus of this study is steganography, particularly linguistic steganography. Linguistic steganography is one of the more challenging forms of covert communications to detect due to its robust nature, low payload and high imperceptibility (Castiglione, et al., 2012; Trovi, ShivarKumar and Das 2016). Deploying a covert channel using spam email can be effortlessly achieved through the use of online tools such as Spammimic.[1] Spammimic is a probabilistic context free grammar (PCFG) that employs linguistic steganographic methods to conceal a secret message within a text corpus that mimics spam email (Castiglione, et al., 2012).

The use of Spammimic, in particular, was emphasised in a murder case investigation where cyber-forensic personnel discovered that a murder was orchestrated by the husband of the victim, who covertly communicated with the killer using stegospam messages generated via Spammimic (Yu, 2015). It is, therefore, conceivable that cyber-criminals and co-conspirators may attempt to conceal their communications by means of encryption. However, if the encryption is too obvious it would arouse suspicion. Consequently, criminals would prefer that their communications go undetected. To this end, spam email offers an ingenious camouflage, because most users including trained digital investigators probably do not pay much attention to spam emails (Yu, 2015).

The paucity of research committed to the detection of stegospam files, as well as the inherent vulnerability, ubiquity, and presumed innocuity of spam email has presented cyber-criminals with an ideal medium of cover to effortlessly establish channels for covert attacks and communication. Consequently, a cogent approach capable of detecting such forms of steganographic communication is required. This research addresses the problem by proposing a steganalysis approach to effectively classify stegospam from non-stegospam files. The research questions that exist to address the research problem are considered next.

[1] *https://www.spammimic.com/*

1.3 Research Questions

This section presents the main research question identified to address the research problem and also lists the secondary research questions that assist in answering the main research question.

1.3.1 Primary research question

The primary research question addressed by this research study is:

Can the implementation of Natural Language Processing methods in a steganalysis approach effectively detect or classify stegospam files created using a probabilistic context-free grammar from non-stegospam files?

1.3.2 Secondary research questions

The primary research question is answered by evaluating the implementation of a steganalysis approach that incorporates Natural Language Processing (NLP) methods, through a series of experiments which are within the context of the primary research question. The following secondary research questions (SRQ) were also considered to determine the metrics of the measurements which are required to answer the primary research question.

SRQ 1. Why is spam email considered an ideal carrier for covert communications?

SRQ 2. Which linguistic steganographic techniques are favoured in order to embed a secret message within plaintext files?

SRQ 3. Which steganalysis techniques are commonly employed for the detection of linguistic steganography?

SRQ 4. Which Natural Language Processing methods or algorithms are most effective in detecting or classifying stegospam from non-stegospam files?

This research study proposes an NLP based steganalysis approach for the detection of stegospam created using a PCFG from non-stegospam files. To respond to the research question the proposed steganalysis will be assessed against its ability to analyse and process large volumes of data; mine deep features from the input dataset; and for its ability to

efficaciously classify stego from non-stegospam files using various classification or machine learning methods.

The next section describes the research objectives.

1.4 Research Objectives

To respond to the problem statement by answering the research questions presented in the preceding section, the research objectives will be considered next.

The objective of this research study is to answer the research question by evaluating current steganographic methods used to conceal a secret message within different cover mediums in particular text covers, current steganalysis approaches used to detect linguistic steganography and the various Natural Language Processing (NLP) methods and algorithms in order to construct and evaluate the proposed steganalysis in its ability to method capable of efficaciously detect stego from non-stegospam files or messages.

Hence, the objectives (OB) of this research are:

OB 1. Identify and discuss the vulnerabilities and motivation for the election of spam email as a covert channel.

OB2. Identify existing linguistic steganalysis methods and determine its suitability for the detection of stegospam files generated using a probabilistic context-free grammar (PCFG) from non-stegospam files.

OB3. Identify and describe various Natural Language *Processing* (NLP) *methods* or algorithms that may be considered for the proposed steganalysis.

OB4. Evaluate the ability of a proposed NLP-based steganalysis by conducting several controlled experiments on existing corpora known to contain stego and non stegospam files.

OB5. Identify any limitations not addressed by the existing or the proposed steganalysis, which can be addressed in further research.

1.5 Research Hypothesis

Advancements in the field of NLP – a branch of Artificial Intelligence (AI) −, has seen it implemented across different fields to address various tasks and problems related to:

- sentiment analysis;
- machine learning and translation;
- text categorisation;
- spam filtering;
- information extraction; and
- summarisation (Khurana, et al., 2017).

Natural Language Processing (NLP), employs algorithmic methods entrenched in statistical techniques to determine the semantic sense in textual data. Early applications of NLP were more deterministic in the sense that sample corpora were evaluated against predetermined patterns to complete tasks such as language translation. However, recent developments to NLP are more dynamic in the sense that they implement machine learning techniques. A subfield of NLP is Natural Language Generation (NLG) or the creation of understandable text derived from qualitative or quantitative inputs. These techniques are commonly employed by Probabilistic Context-Free Grammar (PCFG) to auto-generate understandable text (Leeson, et al., 2019).

Subsequently, NLP has been progressively implemented within the cyber-security sector to analyze and pre-empt various types of threats including phishing attacks by performing Latent Semantic Analysis (LSA) on email text messages (Peng, Harris and Sawa, 2018).

Advancements in NLP offer several features and benefits to this research study that include:

- advanced data analysis, classification and machine learning capabilities;
- deep feature mining or extraction capabilities;
- rapid data transformation or manipulation on large datasets; and
- improved statistical and empirical analysis.

Evaluating the efficacy of these features in classifying stego from non-stegospam files may be achieved through a series of controlled experiments using an NLP based constructed prototype. Additionally, by implementing the same input dataset and measuring the same metrics across each experiment a meaningful comparison can be established between the different NLP methods used in the steganalysis. The hypothesis, therefore, is:

> **The implementation of Natural Language Processing methods can effectively detect or classify non-stego from stegospam files generated using a probabilistic context-free grammar.**

The research presented in this study is designed to test this hypothesis through a series of controlled experiments and will be discussed in greater detail in Chapter 5 (Research Design and Methodology). This section has presented the hypothesis of this research. The next section will discuss the delineations and limitations of this research.

1.6 Delineations and Limitations

This section discusses the delineations and limitations of this research.

The approach adopted by this research study provides a standard input dataset comprising of both stego and non-stegospam files which were used for training and testing the proposed steganalysis. Considering the research questions and objectives of this study, only the plaintext contents of each file type within the corpus were considered. All metadata, inline multimedia and email attachments were scrubbed and excluded from the input dataset. Nonetheless, steganographers and co-conspirators can exploit the metadata and attachments of an email message to establish a covert channel (Halip, et al., 2012). This is discussed further in Chapter 7 (Conclusion and Future Work).

Additionally, all stegospam messages included in the corpus were generated using the Spammimic steganographic tool. Whilst other linguistic steganographic tools such as T-Lex and NiceText were accessible, they exhibited key drawbacks. Both T-Lex and NiceText use a rudimentary steganographic technique known as ‘synonym substitution’ – which substitutes certain words with those from the same synset (a set of synonyms) to generate the cover text

(Taskiran, et al., 2006). The drawback of this approach is that it does not consider the genre, author style and sentence context of the synonym. Consequently, these tools generated sentences with synonyms that deviate from typical usage rendering them detectable by language models, additionally the cover text generated by both T-Lex and NiceText do not mimic a typical spam message (Taskiran, et al., 2006). These tools are discussed further in Chapter 4 of this research study.

As defined in Chapter 5, all experiments conducted to evaluate the hypothesis were executed using a single stand-alone system running the Microsoft Windows 10 (64 bit) operating system. This was done to ensure a controlled experimental environment and to minimise the effects of any external variables. Additionally, several computing paradigms such as WLANs (Wireless Local Area Network), IaaS cloud services (IaaS – Infrastructure as a Service) and virtualization were considered; however, due to the scale of the corpora and the desire to ensure an isolated and controlled experimental environment the stand-alone system was used.

The learning and classification model is declared as an independent variable for which the Support Vector Machine (SVM) and the Naïve Bayes classifiers were employed. The rationale and benefits of the chosen classification models are discussed in Chapter 4 – Natural Language Processing. Other learning algorithms such as k-Nearest Neighbour and Artificial Neural Network (ANN) were considered but not implemented due to the manner in which these machine-learning algorithms learn and classify data. This is further addressed in Chapter 4 of this research study. The significance of the research is discussed next.

1.7 Significance of the Research

The rapid growth and innovation of the internet has presented numerous opportunities for end-users, businesses and governmental organisations to provide and render services remotely, and to communicate with other users and consumers using a variety of applications. Regrettably, this has also presented opportunities for cyber-criminals to launch more sophisticated attacks on unsuspecting users and organisations (Okutan and Cebi, 2019). Whilst there are several mediums to launch an attack, emailing systems remain a popular

choice for phishing, Business Email Compromise (BEC), spoofing and covert attacks (Adam, 2018; Okutan and Cebi, 2019).

In its early days, BEC attacks usually began with hacking and spoofing of email accounts of CEOs or CFOs of businesses and then demanding funds; however, nowadays these attacks have developed in sophistication predominately on the social engineering front (Adam, 2018). The BEC attacks encapsulate an array of techniques such as spear-phishing, malware injections, Denial of Service (DOS) and steganography (Adam, 2018; Okutan and Cebi, 2019).

Most studies concerned with steganography have a propensity to focus on images and other multimedia file types due to the fact that they offer a higher payload capacity, as opposed to plaintext file types (Ahvanooey, et al., 2019; Castiglione, et al., 2012). Given the ubiquitous nature and the perceived innocuity of spam, it makes spam email messages an ideal medium for covert based attacks (Castiglione, et al., 2012). Consequently, this research study will investigate the threat associated with spam emails as a covert channel through the implementation of a steganalysis that incorporates NLP methods. The knowledge gained may be applied to similar situations for the detection of linguistic steganography through other text-based mediums.

1.8 Chapter Conclusion

This chapter provided a background to the concept of steganography and the research problem statement. Additionally, it discussed the research objectives, research questions, research hypothesis and the rationale for this research.

1.9 Outline of Chapters

The rest of this research entails the following chapters:

Chapter 2 – Implementation and Exploitation of Spam Email. This discusses the email delivery infrastructure, the exploitation and security vulnerabilities associated with spam email, overview of spam filtering techniques and legislative control of spam.

Chapter 3 – Steganography. This introduces and discusses the various forms of steganography, key considerations when choosing a cover medium, linguistic encoding methods and steganalysis, and an analysis on the different cover mediums.

Chapter 4 – Natural Language Processing. This introduces and discusses the various Natural Language Processing concepts and processes. Outlines a typical Linguistic Analysis pipeline and analysis using NLP algorithms.

Chapter 5 – Research Design and Methodology. This provides a discussion the research design and methodology followed in this research.

Chapter 6 – Experimental Results. This presents the empirical observations and results recorded from the research performed.

Chapter 7 – Conclusion and Future Work. This presents the findings, conclusion and future work.

2

Vulnerabilities and Exploitation of Emailing Systems

This chapter presents a discussion on email systems, its benefits, drawbacks and vulnerabilities. Additionally, it highlights and discusses the motivation for the election of spam email as a covert channel. Consequently, this addresses the first research objective (OB1) – to identify and discuss the vulnerabilities and motivation for the election of spam email as a covert channel.

Introduction

To achieve this objective, this chapter examines the anatomy of the email system to ascertain any vulnerabilities associated with it, and presents an analysis of existing spam filtering approaches to determine why is spam email a persistent problem. Additionally, this chapter presents a discussion on the exploitation of spam email and the countermeasures implemented to mitigate them. Finally, it deliberates on the appropriateness of spam as an ideal medium for covert communications, therefore, addressing the secondary research question (SQ1):

Why is spam email considered an ideal carrier for covert communications?

In recent decades, cyber-criminals have employed innovative techniques, particularly spam email, for exploiting email systems in a variety of criminal activities. This is largely due to the fact that email provides convenience and low cost, hence, supporting cost-effective communication amongst criminals. Consequently, email has emerged as an additional point of focus when searching for digital evidence in the field of digital forensics (Yu, 2015). However, current cyber-security and forensics research focus essentially on the analysis of legitimate email. Very few, if any, focus on the one email source that may contain significant

information about criminal activities or potential attacks in spam email. This is fundamentally attributed to the fact that spam email is largely presumed innocuous (Yu, 2015).

Such an inherent perception of the innocuity of spam email favours cyber-attackers (including those with moderate computing skills) to exploit such vulnerabilities. Current research on spam email largely focuses on spam detection employing a variety of spam filtering techniques (Liu & Wang, 2011; Yu, 2015). This is unrelated in the detection of concealed information within the spam email file or message itself which is more aligned to digital forensics. Spam filtering methods merely assist in reducing false positives, rather than unearthing messages hidden in spam emails (Yu, 2015).

Consequently, low-cost, ease of use, and its presumed innocuity have been the driving forces and motivation behind the use of spam email in numerous criminal activities and continues to do so. Such cases discussed in this chapter include the concealment of incriminating data, the establishment of covert channels between co-conspirators by concealing secret messages in spam email, and the use of spam email as a form of distraction.

This section provides an introduction to this chapter. In the next section email systems are defined and discussed. For the purpose of this research study the term 'spam' implies 'spam email', consequently these terms are used interchangeably.

The structure of this chapter is as follows:

Section 2.1 - Defines and provides a discussion on the history of electronic mail (Email).
Section 2.2 - Presents and discusses the various components of the email delivery system.
Section 2.3 - Discusses the Simple Mail Transfer Protocol and its vulnerabilities.
Section 2.4 – Definition of Email Spam
Section 2.5 - Presents a detailed discussion on the positive and negative impact of spam.
Section 2.6 - The different categories of Spam are presented and discussed.
Section 2.7 - Spam filtering techniques are presented and discussed to ascertain why spam email is still prevalent.
Section 2.8 - Presents a discussion on different cyber-crimes perpetuated using spam email.
Section 2.9 - Conclusion of spam email as an ideal medium for covert communications.

2.1 History and Background of Electronic Mail

Developed by Raymond Samuel Tomlinson in 1972, emails remain one of the most ubiquitous forms of electronic communication (Gibbs, 2016). Electronic mail (email) by definition refers to the "distribution of electronic messages from one computer user to one or more recipients via a telecommunication network" (Brush and Ferguson, 2019). Early implementations of email systems were purely based only on the American Standard Code for Information Interchange (ASCII) encoding system which allowed users to create plaintext email messages only. However, with the introduction of the Multi-Purpose Internet Mail Extension (MIME) designed specifically for the Simple Mail Transfer Protocol (SMTP), users were able to exchange different kinds of data files over the Internet such as images, audio, and video (Brush & Ferguson, 2019; Jithin, 2016).

Used worldwide, email facilitates both professional and personal communications. It offers rapid and free communication between users over varied distances and requires very little computing skill to be able to send or receive an email message. Today, users can send and receive emails from most internet capable devices in an instant. According to the Email Statistics Report 2020-2024 by the Radicatic Group an estimated 306 billion emails are sent and received daily from an estimated 5.5 billion email accounts worldwide; this therefore affirms that email remains one of the preferred communication mediums and a major contributor to internet traffic.

In the business and commercial sectors emails form an integral method of communication for Business to Business (B2B) or Business to Customer (B2C) communications that is reliable, cost-effective and accessible (Brush and Ferguson, 2019). Additionally, emails are non-repudiative which implies that a sender cannot refute sending an email message, a vital component in business and financial transactions. However, hackers and spammers employ several techniques to mask their origin (as discussed later in this chapter). Another major advantage of email communication is that it offers an asynchronous delivery system which implies that the recipient does not need to be online for the sender to send an email (Clark, et al., 2018; Wozniak, 2004).

Whilst email presents a host of benefits there are notable disadvantages which are inherited with this technology; these include:

- the proliferation of spam emails - unsolicited bulk email (UBE) which can flood a user's inbox a Denial of Service (DOS) attack;
- the distribution of email based viruses such as the notorious Melissa virus − a fast-spreading macro virus that is distributed as an e-mail attachment;
- the rapid diminution of disk storage space due to email attachments and phishing and spear-phishing attacks − a criminal approach that employs both social engineering and technical trickery to elicit personal identity data and financial credentials from unsuspecting users.

This is achieved through the implementation of email known as 'email spoofing' that purports itself to be from a trusted source (Brush and Ferguson, 2019; Gangavarapu, Jaidhar and Chanduka, 2020; Rouse, 2005).

To provide some context for the examination on the vulnerabilities associated with email systems, we need to understand the following:

- how an email system operates;
- discover why it is insecure; and
- how criminals leverage such insecurities.

This is addressed in the next sub-section.

2.2 Email delivery infrastructure

What are the mechanics of an emailing system and why it is susceptible to spam? To answer these questions, it is necessary to understand its delivery infrastructure. Doing so, will provide a clearer understanding of how and why spammers or other forms of cyber-criminals exploit this medium. Additionally, it will broaden an understanding of the strengths and weaknesses associated with spam filters.

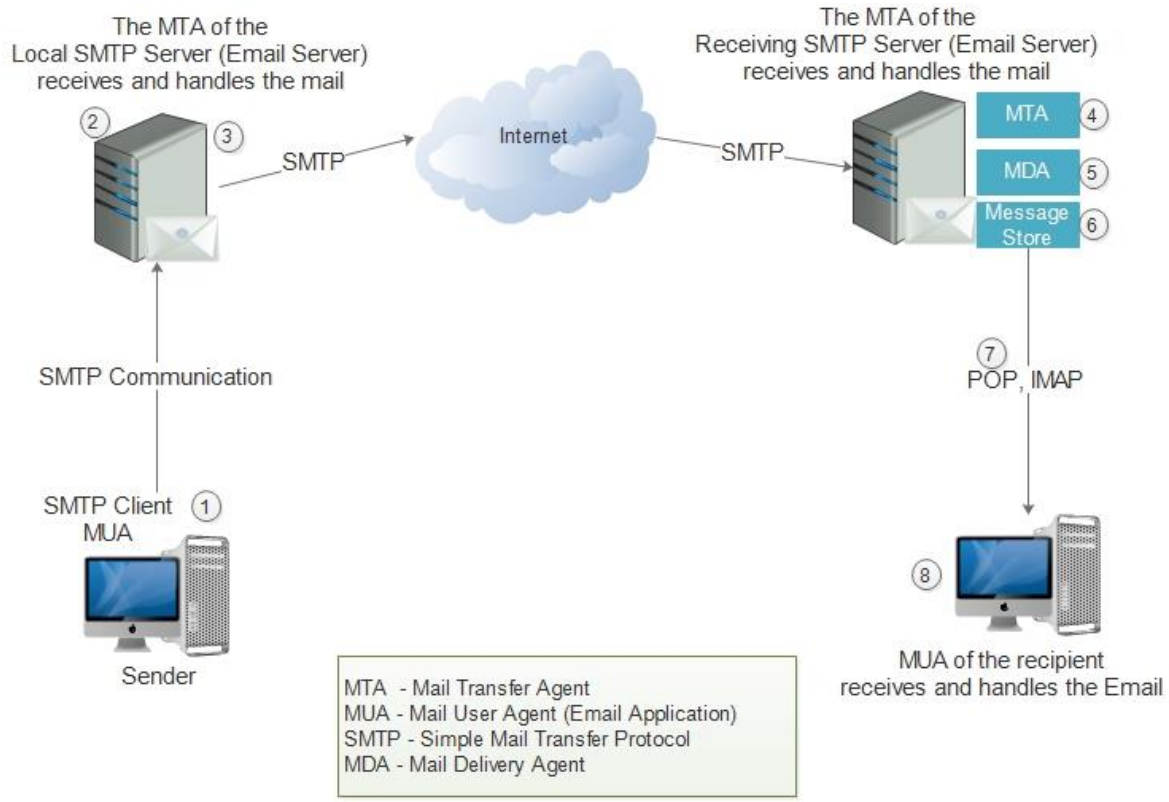
Firstly, email is a simple messaging-based communication system that requires very little skill in order to use. It was not originally developed to handle security issues like authentication,

data integrity and encryption (Brush and Ferguson, 2019; Schryen, 2007). Users generally compose an email intended for a particular recipient, attach any files/documents, enter the targeted email address and then click on the send button. The process and infrastructure that supports the delivery of the email is abstract to the end-user but not to the spammer or the attacker. To understand how spam works we need to examine an emailing system and examine its various components in particular the Simple Mail Transfer Protocol (SMTP).

A standard emailing system has three major components which include:

- ***Mail User Agent*** *(MUA)* – such as Microsoft Outlook or Gmail (basically an email application).
- ***Message Transfer Agent*** (MTA) – refers to a software that is responsible for transferring an email message from the sender to the email server/recipient using the Simple Mail Transfer Protocol (SMTP).
- ***Post office Protocol*** *(POP)* ***and Internet Mail Transfer Protocol*** *(IMAP)* ***Servers*** - which is used to receive or view emails stored on the local SMTP server or Email server.

The function of each component and their dependency on one another is illustrated in Figure 2.1.



Source: (Schryen, 2007)
Figure 2.1 – Emailing System/Process

In Figure 2.1 the process of sending or receiving an email entails the following steps:

- **Step 1:** The End-User uses a Mail User Agent (MUA) which is essentially a software or program such as Microsoft Outlook or a Web based agent such as Gmail to compose an email message. Here the destination address, carbon copy, title of the email and body is defined. At this point the MUA is regarded as the Simple Mail Transfer Protocol (SMTP) client. Once the mail has been composed the user clicks on the send button to send out the email using the SMTP protocol (Schryen, 2007).

- **Step 2:** The Mail Submission Agent (MSA), a sub process of the Mail Transfer Agent (MTA) which is the backbone of the email process, delivers the email from the MUA of the sender to the MTA of the SMTP server. The mail is then stored on the Local Message Store of the SMTP server. Here additional tasks such as virus scanning and spam mail filtering can be performed (Schryen, 2007).

- **Step 3:** The SMTP server uses the email address of the recipient and queries the Domain Naming Server (DNS) for the destination Internet Protocol (IP*)* address of the recipient. Once resolved the email is then routed via the internet until it reaches the destination network of the intended recipient (Schryen, 2007).

- **Step 4:** Once the email reaches the targeted network the MTA of the SMTP server receives the email and stores it on it locally on SMTP Server itself awaiting delivering to the recipients MUA.

- **Step 5:** The Mail Delivery Agent (MDA) a sub process of the MTA that transfers the mail to the message store of the STMP server awaiting delivery to the recipient.

- **Step 6:** The message is stored in the message store or the server inbox awaiting delivery to the MUA of the recipient.

- **Step 7:** The recipient uses a MUA to download the mail stored in the message store by using one of two protocols; Post Office Protocol version3 *(POP3)* or Internet Message Access Protocol version4 (IMAP4). If the MUA uses POP3 the mail is delivered to the recipient and removed off the message store of the SMTP server. If the MUA uses

IMAP4 the mail is delivered to the recipient but a copy of it remains in the message store of the SMTP server (Schryen, 2007).

- **Step 8:** Finally, the recipient's MUA queries the local SMTP Server for any emails. The MUA uses either the POP3 or IMAP4 protocol to receive any new email messages (Schryen, 2007).

As illustrated in Figure 2.1 and discussed in the steps above, SMTP is the core protocol used in the email delivery process, hence, it is appropriate to examine for this research study. Conversely, this will not disclose if SMTP has the ability to detect embedded data in spam messages or not. It will, however, reveal why email systems are inherently insecure and identify grounds for the election of spam email as an ideal carrier for covert communications.

2.3 Simple Mail Transfer Protocol (SMTP)

The SMTP was designed as a transport protocol for electronic mail transmission. It is a text-based protocol defined in 1982 by RFC 821 and updated in 2008 by RFC 5321 to Extended SMTP and has since rapidly become an internet and email standard typically used to send emails to a mail server on Transmission Control Protocol (TCP) port 25 (Klensin, 2008). It is important to note that SMTP only defines the message transport data or metadata which includes, amongst others, the Sender Address, Recipient Address, Return-Path and Date, which forms part of the mail header and not only the message content itself (Klensin, 2008).

Based on the objective of this research study we only consider the body of spam messages for the detection of steganographic (hidden) content. Consequently, all metadata (which includes several SMTP commands) will be scrubbed from the input corpora. Therefore, a discussion on SMTP is necessary to identify any vulnerabilities that may serve as motivation for spammers and steganographers alike.

2.3.1 Vulnerabilities of the Simple Mail Transfer Protocol (SMTP)

The Simple Mail Transfer Protocol (SMTP) was designed to transport and deliver email messages reliably and efficiently. Mail clients such as Microsoft Outlook and Mac OS XMail implement the SMTP protocol to transfer email messages to a local mail server, typically referred to as an SMTP Server. The local SMTP Server, in turn, uses the SMTP protocol to relay email messages across an internetwork until to reaches the intended receiving mail server as illustrated in Figure 2.1. While the SMTP is extremely crucial in the email delivery system; however, it is intrinsically insecure and does not provide any degree of encryption or security between the SMTP clients or the servers. The original relaying model of STMP was exclusively designed around the concept of 'cooperation' and 'trust' between the SMTP servers. Owing to its ease of use and inherent lack of security, attackers can effortlessly impersonate an SMTP server and negotiate directly with neighbouring SMTP Servers to relay mail objects that can deceive naïve users into trusting that they have originated from a legitimate source (Clark, et al., 2018; Klensin, 2008).

Additionally, SMTP operates at the transport layer of the TCP/IP Model (which is responsible for logical transmission and communication between processes) it does not offer any built-in authentication services (Klensin, 2008). Email security, however, can be achieved through the implementation of an end-to-end encryption protocol such as Secure/Multipurpose Internet Mail Extensions (S/MIME) or any other security extensions such as Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM). The S/MIME, however, only authenticates communication between two neighbouring servers as opposed to a chain of interconnected SMTP servers with significantly less authentication services rendered between mail user agents (MUA) and SMTP servers. This makes SMTP intrinsically weaker than most end-to-end systems that implement digitally signed messages (Klensin, 2008).

The inherent lack of security of the SMTP protocol is a significant motivator for spammers and attackers inclined to deploy an array of cyber-attacks such as phishing, spoofing, Business Email Compromise (BEC) and ransomware as they can remain completely anonymous through the implementation of an anonymizer and imperceptible to detection (Brush and Ferguson, 2019; Chow, 2012). Additionally, this makes spamming effortless, affirming spam as an ideal channel for covert communications.

2.3.2 Exploitation of VFRY and EXPN

From the preceding subsection, therefore, it is clear that intrinsically SMTP is insecure. Furthermore, it operates on a 'trust' system. However, to realise the research objective 2 (OB2) - ***'Identify and determine the vulnerabilities and motivation for the election of spam email as a covert channel'*** a further examination of SMTP is needed, in particular its commands.

For a communication medium to be considered a covert channel it must be cost effective, ubiquitous, easy to implement and be imperceptible to unauthorised observers (Clark, et al., 2018; Yu, 2015). Consequently, this subsection will consider the exploitation of two commonly used SMTP commands namely; VFRY and EXPN.

The VRFY command is an SMTP request sent to an SMTP server to verify an email address whilst EXPN queries the recipient of an email message to confirm that the argument identifies a mail list and, if so, return the membership of that mail list. During internal audits these commands are useful in tracking email communications. Externally, however, spammers exploit these commands to harvest email addresses and allow hackers to learn about valid usernames or email addresses on a targeted system or domain. With such information at hand, spammers can masquerade as a valid user or send out spam using a valid email address (Brush and Ferguson, 2019; Chandal, 2017).

Whilst a logical solution would be to implement a third-party application on an SMTP server or for a client to assist in the detection of invalid or suspicious email addresses, this would in practice, contradict the internet mail delivery principle that if a message can be delivered it should be delivered − regardless of syntax errors and content. Alternatively, if a message is rejected because it was found to contain harmful content, it still remains the decision outside the scope of the SMTP protocol and, by definition, the SMTP server. Consequently, making email a sustainable solution for spamming (Klensin, 2008). This further emphasises the vulnerabilities of the SMTP protocol and affirms why email systems remain a preferred medium for an array of attacks. Additionally, it goes some way in addressing the secondary research question (SQ1):

Why is spam email considered an ideal carrier for covert communications?

2.3.3 Securing Email Traffic

In this section the different approaches adopted to secure email systems and to detect any limitations are discussed. The rationale for this discussion is to highlight the fact that regardless of the approach, they are fundamentally optional and serve as add-ons to the SMTP protocol, and as such, present opportunities for exploitation due to the lack of or the misconfiguration of these add-ons.

As stated above, the email system was designed without security in mind and despite its ubiquity it remains fundamentally insecure. Email messages are constantly transmitted across insecure, public connections which lie beyond the organisation's security boundary. When email messages lack suitable security they become vulnerable to interception and can be easily copied or modified. Securing an emailing infrastructure is the responsibility of an organisation's IT department, particularly its cyber-security personnel (Clark, et al., 2018; Stine and Scholl, 2010).

Email archives are characteristically extensive, stored in plaintext and highly susceptible to hacking. Spam and malware are prevalent, though filtered by many email providers. Phishing, Whaling[2] and spamming remain persistent problems (Clark, et al., 2018; Stine and Scholl, 2010). To address this problem, several security solutions have been introduced such as Pretty Good Privacy (PGP), Secure Multipurpose Internet Mail Extensions (S/MIME), Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM). However, due to a lack of proper administrative involvement and supporting IT infrastructure, they are commonly misconfigured or lack adoption thereof (Orman, 2015; Särud, 2016; Seltzer, 2013).

Major email application developers such as Microsoft and IBM implement S/MIME to secure their emailing products which are used by organisations where there is adequate motivation to secure their intellectual property or to comply with governmental regulations, whilst general users have limited options. The PGP has always been the preferred option to secure email for many users but it has been beleaguered by the lack of adoption and severe usability concerns surrounding key management. Consequently, users and experts have abandoned it, including Phil Zimmermann the creator of PGP (Clark, et al., 2018; Stine and Scholl, 2010).

[2] *A whaling attack is a form of phishing attack directed at the CEO or high profile executives of a company.*

Despite the implementation of S/MIME by email vendors, email messages are rarely transmitted securely across an encrypted connection and has become a common vector for pervasive cyber-attacks including message forgery, spam, malware and phishing (Brush and Ferguson, 2019; Clark, et al., 2018).

Securing email messages on an external internetwork is extremely challenging as opposed to stand-alone computers or an internal network which usually lie within the security parameters of an organisation. In a network infrastructure, data resides in two main areas: one is on a data store which is a local database; and the other is in transit i.e. when data is being transmitted on the wire. When stored on a local database, data can be easily encrypted and secured. The same is true for email messages stored in a local mailbox. The threat arises when the message has to leave the sanctuary of the data store and is transmitted across the Internet. Whilst other solutions such as SPF and DKIM have been implemented, they are not without their vulnerabilities as discussed in the following sections.

Considering that SMTP is the transport protocol for email messaging it renders them highly vulnerable to interception and enumeration techniques as all messages are transmitted in clear text. This vulnerability motivates attackers to opt for email systems as a popular medium for the distribution of malware, spam and phishing attacks. Consequently, emails have rapidly become a common entry vector for attackers seeking to gain access to an enterprise network or elicit classified information (Brush and Ferguson, 2019; Lord, 2018).

Any attempt to successfully secure an email transmission requires numerous variables to be in place including all communicating parties that need to know in advance of each other's ability to engage in a secure channel. Even if this is possible, SMTP neither offers any inherent facilities for the authentication of the sender, nor does it validate if an SMTP Server is an authorised server, which could result in email spoofing and more commonly spam email. Hence the establishment and investment of domain authentication by email providers has been implemented to try and minimise domain and email spoofing (Clark, et al., 2018).

Securing an email system is a demanding task. Not only is the universal deployment of a secure email system intangible, numerous stakeholders around the world are engaged in an assortment of different solutions rendering it problematic for a single implementation to

emerge. The Sender Policy Framework (SPF) and the DomainKeys Identified Mail (DKIM*)* are two such solutions (Clark, et al., 2018) and are described below:

- **Sender Policy Framework.** Spear-phishing and whaling are forms of phishing attacks which encompasses the distribution of fake emails by spoofing or impersonating a sender. Consequently, SPF was developed to help identify and eliminate such forms of attacks through an email authentication process. The SPF allows the recipient of an email message to validate a sender's identity but at an organizational level[3] in order to detect forgery and to prevent spam (Grayson, 2020; Weiss, 2018).

  To implement SPF, the Domain Administrator of an organisation must publish the SPF policy defining which mail servers are authorised to send emails from its domain. This information is made available in an SPF.txt record in their Domain Name Server (DNS). When an inbound mail server receives an incoming email, it references the SPF record of the originating domain to decide whether to accept, reject or flag the message (Dubrovin, 2017: Weiss, 2018). Whilst SPF is a noteworthy authentication protocol implemented across the email industry it remains an optional add-on to SMTP hence not all mail servers support it. The SPF, however, does come with a few drawbacks which include:

  - Forwarded messages typically fail SPF authentication because the forwarder's SPF record does not contain the original senders authorised IP addresses.
  - Many organisations fail to keep their SPF rules and records up to date which result in inconsistencies in the verification process and diminishes its efficacy.
  - The SPF verification is performed on the 'Mail From' parameter found within the email header, which is not visible to the recipient. This means that a fraudster can pass SPF authentication verification for a domain completely unrelated to the sending domain they are spoofing (Dubrovin, 2017).
  - In multi-tenanted email environments, IP space across consumers is generally shared and consequently, SPF records are also shared. This renders it impossible for SPF to be used to effectively to differentiate between senders on the same hosted platform (Dubrovin, 2017; Särud, 2016).

[3] *Refers to the verification of an email sender against a list of mail servers authorized by an organization to send emails from that domain.*

- **DomainKeys Identified Mail.** Owing to the fact that the original developers of SMTP failed to include any method for authenticating the sender of an email message, any significant changes thereof would have to be considered as optional or add-ons to the SMTP protocol. Consequently, a lot of research and work has gone into addressing this problem and approximately 10 years ago *Yahoo!* and *Cisco* introduced the DomainKeys Identified Mail (DKIM) encryption method (Seltzer, 2013). The DKIM allows the recipient of an email message to verify the domain from which it has originated.

  When a mail server configured with DKIM sends an outbound email, the mail server inserts a digital signature into the message header using its private key known as the *'*DKIM-Signature'. When the recipient receives this email message it issues a DNS-request to the originating domain and by doing so retrieve the public key in order to verify that the signature is valid and that the signed parts of the email has not been modified in transit (Särud, 2016; Seltzer, 2013).

  Whilst DKIM assists in addressing the problem of domain spoofing it does have a few notable drawbacks which include:

  - The DKIM signature verification are mostly supported by major Internet Service Providers (ISP) such as Yahoo!, Microsoft and Google. Not all ISPs implement DKIM.
  - The DKIM is not a built-in feature of SMTP and hence is optional.
  - The DKIM is susceptible to replay attacks because DKIM only signs specific parts of an email message. Hence the message can be strategically modified and forwarded on by an intermediary and the digital signature will still match hence deceiving the DKIM engine or the user (Seltzer, 2013).

In this section different approaches related to the security of an email system were presented and discussed particularly in the public domain and their limitations. The discussion affirms that any solutions recommended are fundamentally optional due to the intrinsic lack of security associated with the SMTP protocol. Additionally, many solutions such as SPF and DKIM require administrative intervention to ensure their efficacy. Based on the statistics

report illustrated in the Graph 2.1 of Spamcop Statistics Report June 2018 - July 2019 an average of 2.7 spam messages are sent per second world-wide, with an estimated 85% of all emails sent being spam. This demonstrates the motivation and persistence for the election of email systems to deploy various forms of cyber-attacks.

Bearing in mind that this research study focuses on the manipulation of spam and not merely normal email messages, a discussion on spam is necessary. Consequently, in the following sections spam email is examined to determine its suitability as an ideal channel for covert communications i.e. steganographic attacks.

2.4 Definition of Spam Email

Prior to addressing the impact associated with spam email it needs to be clearly defined. The establishment of a definition for spam email requires a clear examination of its characteristics which is imperative to this research and will serve as a benchmark. The underlining characteristics of spam email are:

- it is always unsolicited – the recipient has not granted permission to receive it;
- it is typically commercial or promotional in nature;
- it is usually sent in en masse; and
- it generally contains a link redirecting the recipient to a website or webpage (Harris, 2003).

Spam Track at the Text Retrieval Conference (TREC) defines spam email as *"Unsolicited, unwanted email that was sent indiscriminately, directly or indirectly, by a sender having no current relationship with the recipient"* (Cormack and Lynam, 2006, p91). Consequently, a tentative and generalised definition of spam email would be – an unsolicited and undesirable email message containing promotional or a commercial payload, sent in bulk by a spammer who has no relationship with the targeted recipient(s). Whilst this definition attempts to encapsulate key factors that would characterise spam email throughout this research, it does not declare that it is mandatory for spam email to incorporate a hyperlink for redirection to a website for it to be classified as spam (Schryen, 2007). The next section provides a discussion on the impact of spam email.

2.5 Impact of Spam email

2.5.1 Positive impact of Spam email

Since the inception of email technology, spam has always been in existence. As previously discussed, the rationale behind the use of spam is that it offers an economically viable medium for commercial advertising. Spammers (individuals or organisations who send out spam*)* harvest email addresses from numerous sources including those publicly available. Even if only a fraction of the recipients of a spam message respond to an advertised product or service the sender profits and the spam problem is perpetuated (Cormack, 2008).

It is currently estimated that 85% of all email messages sent worldwide are believed to be spam with an average of 2.7 spam messages sent per second based on the report generated by Spamcop[4] illustrated in Graph 2.1 of the *Spamcop Statistics Report June 2018 – July 2019.* Email marketing has continued to provide great benefits to marketers, from building trust to providing some of the best ROIs of any digital channel (Dormer , 2020). According to the 2020 Email Marketing Trends Survey more than 64% of businesses claim that email-based marketing showed a significant increase in sales (Bujang and Hussin, 2013; Dormer, 2020). As an effective marketing tool, spam email yields a high Return On Investment *(ROI)* for marketers (Brush and Ferguson, 2019). Other benefits include:

- ***Scalability*** – it can reach a much larger audience and can span several countries.
- ***Personalisation*** - marketers can customise an email based on the users' online activities and searches.
- ***Increased Sales*** - it allows a vendor to access a much larger audience and with just a mere 1% success rate on that audience could result in significant profits.
- ***Measureable*** – marketing campaigns can be evaluated based upon target responses.

Given these benefits, marketing a product, service or even promoting a campaign is made much faster, flexible and offers a cost-effective opportunity of reaching new customers or retaining existing ones (Brush and Ferguson, 2019; Dormer, 2020).

[4] *https://www.spamcop.net*

In recent years the inclusion of videos in emails have proven an effective marketing strategy, affording companies the ability to deploy product demos and customer testimonials, creating customised communication between businesses and consumers. This is further driven by the fact that email is increasingly accessible on heterogeneous devices and the migration of on-site mailboxes to cloud mailboxes, creating opportunities for smart marketers (Bujang and Hussin, 2013; Gurin & Fisher, 2013).

The Radicati Group Report[5] for 2018-2022 shows a 20% increase of all email generated revenue for the 2019 period and an estimated growth of 17% per year until 2020 as shown in Table 2.1 below, in the World-Wide Email Forecast, 2018-2022. This demonstrates the positive impact and significance of email-based marketing campaigns to an organisation's profit line and potential to yield better Return on Investment (ROI).

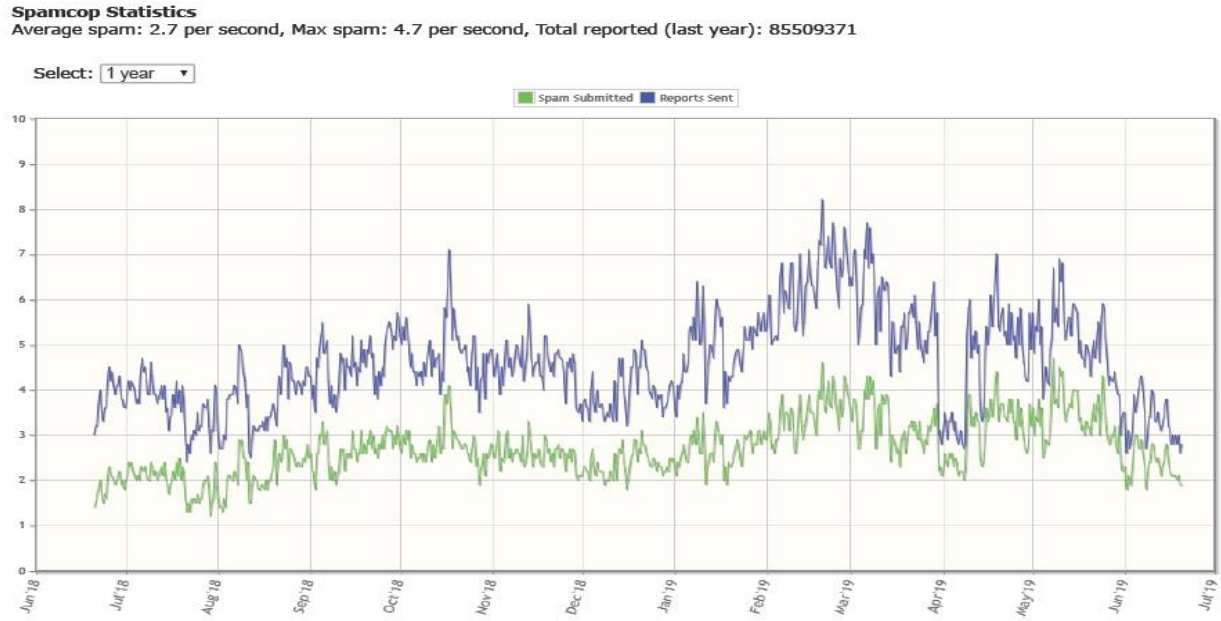
Worldwide Email Market Forecast	**2018**	**2019**	**2020**	**2021**	**2022**
Worldwide Email Users (M)	3,823	3,930	4,037	4,147	4,258
% change		*3%*	*3%*	*3%*	*3%*
Worldwide Email Market Revenues ($M)	**$33,128**	**$39,633**	**$46,941**	**$55,142**	**$64,269**
% change		20%	18%	17%	17%

(*Source: The Radicati Group - Email Market Report - www.Radicati.com).*
Table 2.1 – World-Wide Email Forecast, 2018-2022

In Figure 2.2 below, the graph of the *Spamcop Statistics Report June 2018-July 2019* - illustrates the number of spam emails detected and reported per second between June 2018 – July 2019. This affirms ongoing spam trends despite attempts to mitigate them.

The rationale for the presentation of this information is to demonstrate the proliferation and ubiquity of spam email, a key requirement for any elected covert channel.

[5] *https://www.radicati.com/wp/wp-content/uploads/2018/05/Email-Market-2018-2022-Executive-Summary.pdf*



(Source: https://www.spamcop.net)

Figure 2.2: Spamcop Statistics Report June 2018-July 2019

2.5.2 Negative impact of Spam email

Whilst the motivation behind spam is to deliver marketing content relating to a product or service directly to the user's mailbox, also known as 'direct marketing', it can also be manipulated and used as bait for fraudulent schemes, promotion of a cause, or contain camouflaged links that seem to be for legitimate websites but in fact lead to phishing web sites or sites that host malware (Bujang & Hussin, 2013; Cormack, 2008). *"In fact it has been demonstrated that virus writers hire spammers to disseminate their so-called malware"* (Puertas, Hidalgo and Perez, 2008, p45-112).

Spam email is also widely used in phishing and spear-phishing attacks. According to Tamir, (2013):

> *"Spear-phishing is one of the main tools used by attackers to compromise endpoints and gain a foothold in the enterprise network. The attacker utilizes a specially crafted email message that lures users to perform an action that will result in malware infection, credentials theft or both*".

Spam is increasingly sent from computers infected with viruses. Furthermore, virus developers and spammers are merging their efforts to compromise innocent computer users' systems and convert them into spamming drones or zombies. These malicious programs

spread rapidly and generate considerable amounts of spam masquerading as legitimate sources (Bujang and Hussin, 2013; Clark, et al., 2018).

The effects of spam email are also experienced by network and email administrators who have to employ considerable amount of time and effort in deploying solutions to combatting it. It is currently estimated that spam is accounted for 85% of email traffic based on the SpamCop.net Statistics[6] report. Spam is not only dangerous or a waste of time but also quite disturbing. Receiving unsolicited messages is a privacy violation and often forces the user to see strongly unwanted material, including pornography (Puertas, Hidalgo and Perez, 2008). Increases in spam traffic tend to cause high network congestions resulting in network latency and eventually a Distributed Denial of Service (DDOS). To circumvent this Internet Service Providers and Email Service Providers have to invest heavily in better network infrastructure and storage facilities (Bujang and Hussin, 2013).

To help regulate spam email the American government sanctioned the CAN-SPAM Act in 2003 which establishes a framework of accountability and control over spam. Other governments have also implemented similar acts including the South African government's ECTA[7] (Electronics Communications and Transactions Act of 2002). Sadly, these Acts are largely ineffective particularly if the sender is untraceable (Michalsons, 2008; Muris, et al., 2004).

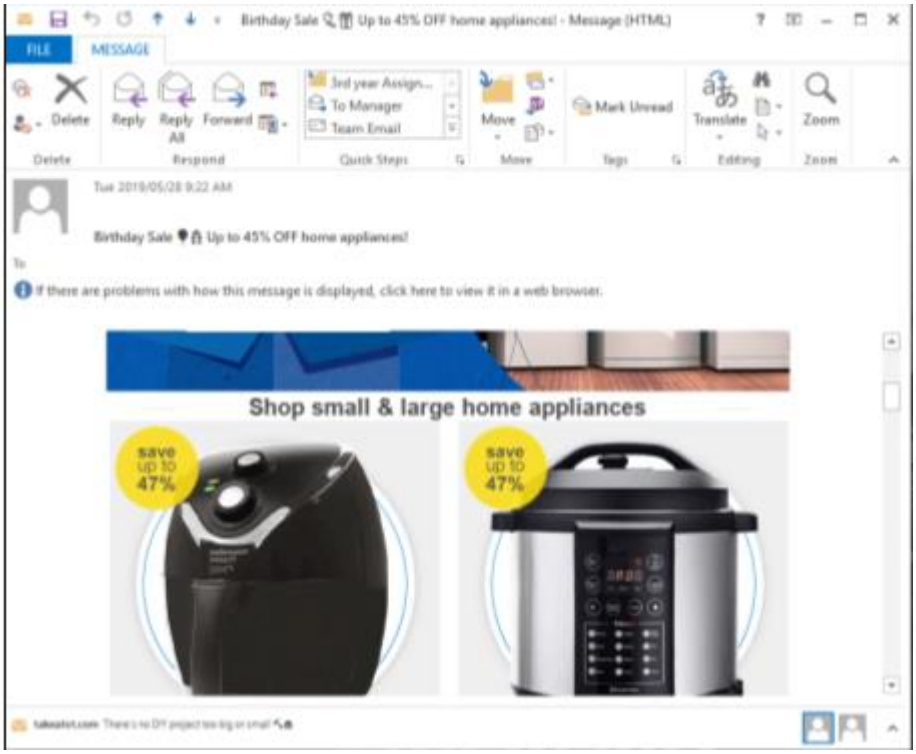
2.6 Categories of Spam email

2.6.1 Unsolicited Commercial Email (UCE)

Unsolicited Commercial Email (UCE) - is a term used to describe electronic promotional messages sent to potential consumers without the consumers' consent or request with the key purpose of providing a commercial advertisement or promotion of a commercial product or service, an example of which is shown in Figure 2.3.

[6] *https://www.spamcop.net*

7 https://www.gov.za/sites/default/files/gcis_document/201409/a25-02.pdf



Source: www.takealot.com

Figure 2.3 – An example of an Unsolicited Commercial Email (UCE)

Whilst this kind of promotional or marketing stream encourages free enterprise and reduces the restraints of trade due to its ease of use and negligible costs, a few specific problems do exist such as:

- Misleading content and heading information.
- Economic and performance liability to the internet, mail servers and enterprise networks caused by the huge volume of commercial spam being sent.
- Costs and frustration imposed on the user to implement spam detection and prevention mechanisms (Rouse, 2010).

The need for commercial based spam has been widely accepted as a cost effective medium for easy and sustainable advertising for many companies. Various jurisdictions such as the US S.877 CAN-SPAM[8] (Controlling the Assault of Non-Solicited Pornography and Marketing - CAN-SPAM Act of 2003) Act of 2003 and more recently Section 45(1) of the Electronic Communication and Transaction[9] Act of South Africa have made provisions for commercial spam but with numerous regulatory requirements that must be honoured for the spam to be

8 https://www.congress.gov/bill/108th-congress/senate-bill/877

9 https://www.gov.za/sites/default/files/gcis_document/201409/a25-02.pdf

considered lawful. One such regulation is that the spam must include a removal link that allows users to remove themselves from a mailing list (Michalsons, 2008).

Unfortunately, based on a study conducted by the Federal Trade Commission in the United States, it was discovered that more than 63% of the sampled UCEs included removal links that did not function. Therefore, similar discrepancies, lack of legislative control, relative anonymity and ease supports the proliferation of spam by making it possible and cost-effective for illegitimate marketers to send spam to billions of email accounts worldwide, while allowing them to conceal their identity or origins of their email messages (Muris, et al., 2004). This further heightens the appeal of spam email as an appropriate channel for covert communications.

2.6.2 Unsolicited Bulk Email (UBE)

This category of spam is a lot more deceptive than UCE as it designed to deceive the recipient into buying something or to elicit sensitive information. Generally, UBE is highly synonymous with virus-laden attachments, chain letters and spear-phishing attacks as opposed to UCE (Rouse, 2010). This is essentially due to the lack of regulation and accountability behind UBE as it is mainly distributed by individuals or shelf companies. Common types of UBE spam include vanity and work on the insecurities of individuals such as weight loss or get-rich-quick programmes (Harris, 2003). Common characteristics of UBE include the following:

- misleading headings used to entice a target;
- false content to motivate the target;
- incorrect contact details; and
- missing or non-functional 'unsubscribe' links hence making it difficult to trace the sender.

The image in Figure 2.4 below illustrates an example of a '419'[10] scam also known as the 'Nigerian Prince' that uses spam emails sent out to numerous unsuspecting users, phishing for a response by scammers which could be a single individual or a group of individuals.

[10] *"419" said as "four one nine" derives from the section of Nigerian law that legislates on con artistry and fraud.*

Why is this message in Spam? It's similar to messages that were detected by our spam filters. Learn more

CENTRAL BANK OF NIGERIA
OFFICE OF THE GOVERNOR
Zaria Street, Off Samuel Akintola
Street,Garki 11, Garki-Abuja.

Our Ref: FGN/CBN/NIG/01/2017.

Your Ref.....

From The Desk Of Mr. Godwin Emefiele.
Governor, Central Bank of Nigeria (CBN)

SUBJECT: Dear Valued Customer.

Dear Friend,

We wish to inform you that your unclaimed payment of USD$10.5 Million in Africa has been released and ready to be paid to you via PREPAID VISA CARD which you will use to withdraw the US$10.5 Million from any ATM Machine in any part of the world.

We have mandated UBA financial advicers Ghana, to send you the ATM CARD and PIN NUMBER which you will use to withdraw all your US$10.5Million Dollars in any ATM SERVICE MACHINE in any part of the world, but the maximum you can withdraw in a day is US$20,000.00 Only.

You are therefore advice to contact the Head of ATM CARD Department of UBA financial advicers Ghana;

Contact Person: Rev. Goodluck Ebola,
Office email address: [redacted]@gmail.com

(Source: https://blog.malwarebytes.com)

Figure 2.4 - Example of an Unsolicited Bulk Email (UBE) 419 scam

In this section the two primary categories of spam were discussed that users are commonly inundated with on a regular basis. The purpose of this section is to demonstrate that whilst spam is marginally permissible within certain regulations, it does not deter spammers and illegitimate marketers from distributing them. Additionally, this section demonstrates the propagation of spam thereby affirming its sustainability and pervasiveness. The next section discusses the different spam filtering techniques.

2.7 Spam Filtering Techniques

For several years since the propagation of spam, email researchers and industry experts have been contributing towards the development and proposal of different anti-spam techniques and algorithms (Bajaj and Pieprzyk, 2014; Puertas, Hidalgo and Perez, 2008. Despite some success in reducing the volume of spam, the problem is still very much prevalent (Bajaj and Pieprzyk, 2014). The fundamental approach adopted by most of these techniques and algorithms were to pre-emptively detect and eradicate spam emails before they reached the end-user (Alkahtani, Gardner-Stephen and Goodwin, 2019; Sanz, Hidalgo and Perez, 2008). To achieve this objective several organisations have opted to install spam filters on peripheral firewalls so that it may help to protect internal email servers and end-users behind it (Christina, Karpagavalli and Suganya, 2010).

The earliest implementation of a commercially based heuristic anti-spam filter was Brightmail in 1999 (Sochor, 2014). This was later followed by the open-source platform named 'Apache' SpamAssassin developed by Justin Mason on the 20 April, 2001 (McDonald, 2004). Quite commonly and widely used, SpamAssassin implements a robust framework which permits plug-ins, allowing for the integration of a wide range of advanced heuristic and statistical analysis tests on email headers and body text, including DNS blacklists, Bayesian filtering, external programs and online databases (McDonald, 2004). Apart from the spam email filtering techniques mentioned above, several others exist such as Naïve Bayes, Support Vector Machines, Neural Networks and Artificial Immune Systems (Bajaj and Pieprzyk, 2014).

Understanding spam filtering techniques requires a general overview of the spam classification process. Hence, the structure of a typical email starts off with the header, the body and any attachment(s). The header is made up of the *Received*, *From*, *Message ID*, *Subject*, *Content type* and *X-mailer* attributes. The body refers to the actual mail content and attachments if any, which represents any file types that is attached to the email. It should be noted that contents of any attachments are not considered for the classification of the email (Bajaj and Pieprzyk, 2014). Prior to textual classification the contents of the email body require some preprocessing which includes Feature Extraction and Feature Selection (Bajaj and Pieprzyk, 2014; Bhowmick and Hazarika, 2016). The underlying objective of pre-processing is to eliminate any data that does not provide useful information regarding the classification of the email object; namely, word redundancy, hyperlinks, images and HTML code (Bhowmick and Hazarika, 2016; Sharma, Rekha and Manisha, 2015).

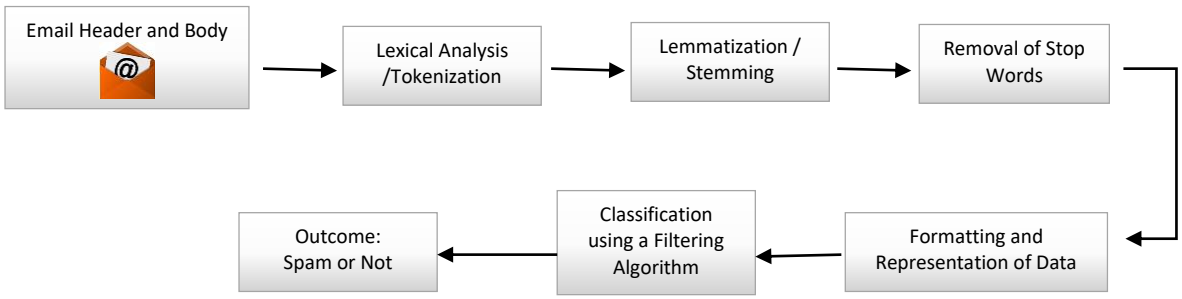
Data preprocessing techniques of email content are discussed in greater detail in Chapter 5 ('Research Design and Methodology').

Preprocessing is achieved via the following steps (Bhowmick and Hazarika, 2016; Guzella and Caminhas, 2009):

- ***Lexical Analysis/Tokenization*** – extracts words from the message body
- ***Lemmatization or Stemming*** – converts words to their morphological base forms e.g. 'running' becomes 'run'

- ***Removal of stop words*** - removal of frequently used non-informative words e.g. 'a', 'the' etc.
- ***Representation*** - converts the email message into a structured format required by the classification algorithm.

The following diagram in Figure 2.5 below illustrates the classification process:



Source: (Bhowmick and Hazarika, 2016; Guzella and Caminhas, 2009)
Figure 2.5 – Email Filtering Process

The tokenization, lemmatization and stemming processes are addressed in greater detail in Chapter 4 (Natural language Processing). In terms of this section, this study remains focused on spam filtering techniques and its efficacy. Whilst various techniques and algorithms have been consistently proposed by researchers over the past few decades, the Naïve Bayes and Collaborative spam filtering techniques have emerged as two of the most effective anti-spam algorithms (Kong, et al., 2006; Christina, Karpagavalli and Suganya, 2010). Therefore, for the purpose of this research the Naïve Bayes and Collaborative filtering approaches are discussed and evaluated.

2.7.1 Naïve Bayes' Classifier

The Bayesian (Bayes') Theorem was developed by Reverend Thomas Bayes in 1761. It *"describes the probability of an event, based on prior knowledge of conditions that might be related to the event"* (Jøsang, 2016, p462-469). The Bayes' Theorem has since been widely implemented across different disciplines ranging from medical science (where it is used to determine the probability of infection) to the philosophy of science (in explaining the relationship between theory and evidence). Since 2000 it has also been implemented in the development of spam filters for email systems in particular Apache SpamAssassin (McDonald,

2004; Schwartz, 2004). Whilst there are numerous applications of the Bayes' Theorem, the scope of this research limits it to spam email filtering.

The classification algorithm or module employed by several anti-spam software including SpamAssassin, SpamBayes and Bogofilter is the Naïve Bayes Classifier (Schwartz, 2004). Patil and Sherekar (2013, p257), defines Naïve Bayes as a *"simple probabilistic classifier that calculates a set of probabilities by counting the frequency of and combination of values within a given dataset".* The Naïve Bayes classifier is a derivative of the Bayesian Theorem and is used in several content-based spam filtering solutions (Rusland, et al., 2017). The fundamental difference between the Bayes' and Naïve Bayes algorithms is that the former uses a sentence or phrase as the known condition such as '*You have won a prize'* in order to determine its probability. Therefore, we can define the probability P of a condition{Spam} for Bayesian as: P{Spam|you have won a prize}. On the contrary, Naïve Bayes algorithm is a lot more granular in that is assumes that each word is independent of the other and then calculates the probability of each word and finally multiplies the possibilities together, therefore we have:

P{Spam|have} x P{Spam|won} x P{Spam|a} x P{Spam|prize} (Stecanella, 2019).

Hence, to calculate the probability of an e-mail being classified as spam or not, the Naïve Bayes classifier uses the Bayes' theorem as shown in Figure 2.6 below.

$$p\,(x|y) = \frac{p(y|x)\,p(x)}{p(y)}$$

Source: (Jøsang, 2016; Patel, 2019)
Figure 2.6 – Naïve Bayes Theorem

Where ***x*** and ***y*** are events and ***p(y)≠ 0***

- **P(x|y)** is a conditional probability: the likelihood of event **x** occurring given that **y** is true.
- **P(y|x)** is also a conditional probability: the likelihood of event **y** occurring given that **x** is true.

- **P(x)** and **p(y)** are the probabilities of observing **x** and **y** independently of each other; this is known as the marginal probability or base rates (prior probabilities).

A further discussion on the Naïve Bayes algorithm is presented in Chapter 4 (Natural Language Processing) for the implementation within the proposed steganalysis.

2.7.2 Vulnerabilities and Efficacy of Naïve Bayes in Spam Filtering

The Naïve Bayes classifier remains one of the most widely implemented probabilistic algorithms for a multitude of server and client-side anti-spamming solutions (Christina, Karpagavalli and Suganya, 2010). However, it is imperative to acknowledge that the quality and precision of a heuristic classifier like Naïve Bayes is subject to several factors including the training dataset used to establish a keyword feature set and the number of attributes referenced during the classification process (Bajaj and Pieprzyk, 2014; Sinclair, 2004).

In a research analysis conducted by Rusland et al. (2017), the Naïve Bayes classifier was evaluated on the properties of Accuracy, Precision, Recall and F-Measure Rates. Rusland et al. (2017) evaluated the Naïve Bayes classifier against two datasets containing spam and ham emails: SpamBase with a corpus of 4601 emails and SpamData with a corpus of 9 324 emails. The results produced when both sets were fixed to the same number of files disclosed that Naïve Bayes achieved a lower accuracy and precision rating of 82.8% when tested using the SpamData corpus as compared to the accuracy and precision rating of 88% when tested using the SpamBase corpus. Consequently, Rusland et al. (2017), demonstrated that the accuracy and precision ratings of the Naïve Bayes classifier was subject to the training datasets from which it learnt. This was further demonstrated by the Naïve Bayes evaluation conducted by Mohammed et al. (2013), who achieved an accuracy and precision rating of 85.9% when tested against the dataset Email-1431.

Consequently, this exposes a vulnerability in the preprocessing phase of the filtering process where the accuracy and precision of the classifier is highly dependent on the quality of the training set. Additionally, if exposed, the training set may be subjected to tokenization attacks which is used to defeat the word classification process by modifying crucial words or features (Bhowmick and Hazarika, 2016; Mohammed, et al., 2013).

Another drawback of a heuristic classifier like Naïve Bayes is the maintenance of its rule set, which can be time-consuming as these rules need to be persistently updated to keep abreast with the latest trends in spamming (Bhowmick and Hazarika, 2016). Even though Naïve Bayes is a fast and accurate spam filtering algorithm that requires a small learning corpus to build a rule, the vulnerability surfaces in the fact that Naïve Bayes uses a resilient independence assumption between features which, in a real world problem, may not be applicable (Bhowmick and Hazarika, 2016; Vyas, Prajapati and Gadhwal, 2015).

Other possible attacks on the Naïve Bayes classifier include 'Letter Substitution', also known as Obfuscation attacks, Random or Common Word attacks, Frequency Ratio attacks and Strong Statistical attacks. A Common Word attack fundamentally encompasses the use of frequently used words that are commonly found in legitimate email and has a low spam probability rate. It was established that the usage of about 50 such words was sufficient to mask a spam email from detection (Sochor, 2014). Random Text attacks, also known as 'Bayesian Poisoning' is a type of statistical attack through which spammers implement cautiously crafted e-mails to attack the core of a Bayesian filter and thus degrade its effectiveness. The addition of carefully selected random text is used to confuse the spam filter into classifying a spam email as a legitimate one. Consequently, increasing the false positive rates classified by the spam filtering method (Bhowmick and Hazarika, 2016).

Additionally, the Naïve Bayes classifier was subjected to further testing by implementing it as a classification algorithm in an Intrusion Detection System (IDS) to aid the detection of a DOS attack imposed by the mass distribution of spam emails. The outcomes revealed negligible improvements in the IDS detection rates within a local area network, with further reduction in detection rates as the capacity of spam increased (Hema and Shyni, 2015).

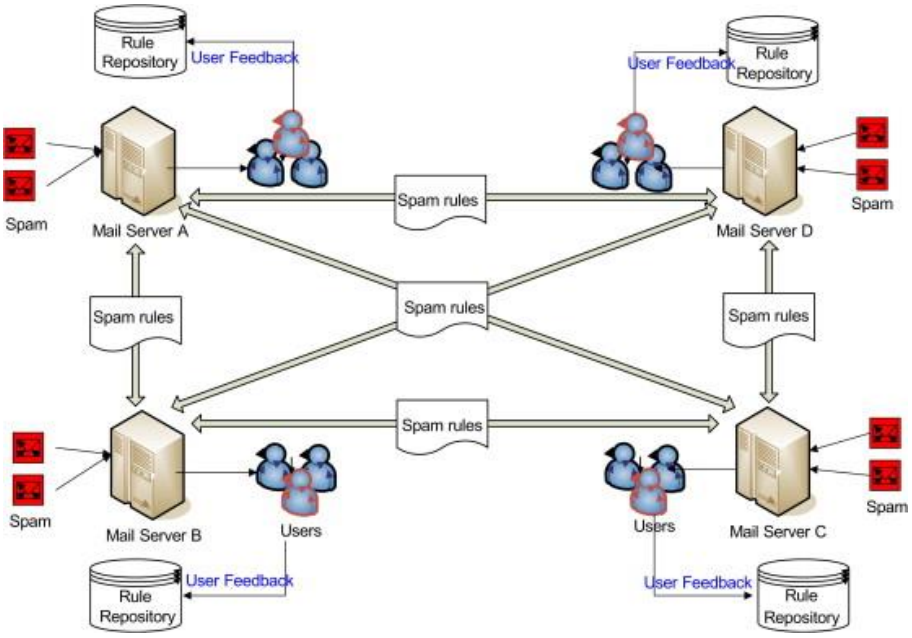
There have been numerous research studies conducted and documented over past few years regarding the efficacy of the Naïve Bayes algorithm, tested against vicissitudes in the datasets or in combination with other algorithms. Ultimately, most demonstrate that the Naïve Bayes classifier has a higher accuracy and precision rates when compared to similar algorithms, which is not to say that it is 'fail proof'. It does generate false positives and negatives and is susceptible to various types of cyber-attacks. Consequently, it presents opportunities for carefully constructed spam messages to serve as ideal carriers for covert communications.

2.7.3 Collaborative Spam Filtering

Content-based filters such as SpamAssassin are referred to as standalone filters since they can only efficiently filter out spam emails based on their locally stored corresponding spam rules, which were constructed using a data mining classifier like Naïve Bayes. However, they are ineffective in filtering out new types of spam emails due to the lack of new corresponding spam rules. Stand-alone filtering necessitates user intervention to remain up-to-date with the latest spam rules. This would require employing someone to consistently evolve the spam rules (Lai, et al., 2009; Wenxuan and Maoqiang, 2013). To address this problem and try to keep spam rules as up-to-date as possible without user intervention, the collaborative spam filtering approach was introduced (Lai, et al., 2009).

Collaborative spam filtering is a more efficient strategy when compared to content-based filtering. In this strategy, the entire collaborative community work together using shared spam knowledge. Therefore, collaborative spam filtering represents a distributed approach to filtering spam, which requires a shared and an efficient database where it can store a users' judgement about which email was classified as spam and which was not: namely, the Distributed Checksum Clearhousing, Vipul's Razor, Pyzor and Cloudmark spam filters.

Additionally, collaborative spam filtering queries various external sources of spam information such as blacklists and whitelists to build and update its own database (Bhowmick and Hazarika, 2016; Wenxuan and Maoqiang, 2013). The collaborative anti-spam solution is constructed on the assumption that all mail servers are in a trusted network and that the rules are exchanged via a secured channel between the servers (Lai, et al., 2009). This is illustrated in Figure 2.7.



Source: (A collaborative anti-spam system - Lai, et al., 2009)
Figure 2.7 – Collaborative anti-spam architecture.

Each mail server in the trusted network must recurrently develop the latest spam rules through the implementation of data mining or machine-learning algorithms. Once the spam rules are generated, it must be synchronised timeously with other mail servers within the trusted network. Therefore, the relevance and quality of spam rules are crucial in the efficacy of the collaborative filtering approach (Lai, et al., 2009). Yielding false positives is a shared drawback of different anti-spam solutions, consequently, it is common practice to combine the opinion of multiple filtering methods before making a decision as demonstrated by collaborative filtering. Whitelists, Blacklists, Challenge-Response Systems and Origin Diversity Analysis are vital sources of spam rules in the collaborative filtering paradigm (Alkahtani, Gardner-Stephen and Goodwin, 2019).

A Whitelist is fundamentally a list of pre-approved or trusted contacts, domains or IP addresses that an email client is allowed to communicate with. A Blacklist, however, is a list of those email addresses, domains and IP addresses that are known sources of spam. Blacklisted sources are blocked at the SMTP connection phase of an email transmission. Common publicly available examples of blacklists include Google Blacklist and Spamhaus (Alkahtani, Gardner-Stephen and Goodwin, 2019; Bhowmick and Hazarika, 2016). Whilst Blacklists are fairly effective, they do have disadvantages. Firstly, Blacklists are maintained by an entity distinct from the user, creating an external dependency for any spam filter.

Secondly, the efficacy of the Blacklist is subject to the methods and update frequency of those who manage it (Alkahtani, Gardner-Stephen and Goodwin, 2019).

Whilst Whitelists place the responsibility of verifying the credibility of the sender on the receiver, the Challenge-Response System transfers this responsibility to the sender. The sender must provide some sort of authentication in order to validate that they are real users. This could be achieved in the form of an automated response that may request or challenge the sender to identify an image or code such as CAPTCHA as shown in Figure 2.8 below, or verify their details via a hyperlink sent to their email inbox. This approach is synonymous in e-Commerce, especially with large online shopping companies like eBay and Amazon (Alkahtani, Gardner-Stephen and Goodwin, 2019; Su and Khoshgoftaar, 2009).

Source: https://en.wikipedia.org/wiki/CAPTCHA
Figure 2.8 – Example of a CAPTCHA Challenge

Like content-based filtering, collaborative-based filter integrates several algorithms to aid in the classification and filtering of spam. Collaborative filtering can be implemented using one of three approaches:

- ***Memory-based Collaborative Filtering.*** This establishes its recommendations by computing the similarity between users or neighbours, i.e. using their past ratings on similar items. It also groups users with similar likes and assumes they are likely to make the same ratings on other items. Thus, it uses an existing user-item database to generate a predication. A common memory based Collaborative Filtering algorithm is the Neighbourhood Collaborative Filtering Algorithm. Memory-based filtering, however, is more susceptible to data sparsity and scalability problems (Fan, 2010; Su and Khoshgoftaar, 2009).

- ***Model-based Collaborative Filtering.*** This builds predication models using machine learning and data mining techniques that allows a system to learn and recognize complex patterns based on the training data. Model-based Collaborative Filtering algorithms include Bayesian models, clustering models and dependency networks. Model-based Collaborative Filtering is less susceptible to data sparsity and scalability problems (Fan, 2010; Su and Khoshgoftaar, 2009).
- ***Hybrid-based Collaborative Filtering.*** This combines collaborative and content-based filtering systems to construct predictions and make recommendations. This approach implements a Content Boosted Collaborative Filtering algorithm which integrates Naïve Bayes as its content classifier and then uses it to populate missing values of a rating matrix hence, creating a pseudo rating matrix. The Hybrid Collaborative Filtering technique is also susceptible to data sparsity and scalability problems much like memory-based Collaborative Filtering (Fan, 2010; Su and Khoshgoftaar, 2009).

Each approach adopts diverse algorithms, yielding different advantages and challenges. An overview is presented in Table 2.2.

CF Categories	Representative Algorithm	Advantages	Disadvantages
Memory-Based recommendations	- Neighbour-based CF with Pearson/vector cosine correlation - Item-based top-N recommendations	- Easy to implement - New data can be added easily - Need not consider the content of items being recommended - Scale well with co-rated items.	- Dependent on human ratings - Performance decrease when data is sparse - Cannot recommend for new users and items - Limited scalability
Model-based recommendations	- Bayesian belief net CFs - Clustering CF - MDP-based CF - Sparse factor Analysis	- Better address the sparsity, scalability and other problems - Improved predication performance	- Expensive model-building - Trade-off between prediction performance and scalability
Hybrid based Recommendations	- Content-based CF - Content-boosted CF - Hybrid CF combining memory-based CF algorithms and model-based algorithms	- Overcome limitations in CF and content-based recommenders - Improve prediction performance - Overcome CF problems such as sparsity and gray sheep	- Have increased complexity and expense for implementation - Need external information that usually not available

Source: A Survey of Collaborative Filtering Techniques (Alkahtani, Gardner-Stephen and Goodwin, 2019).

Table 2.2 – Overview of Collaboration Filtering Techniques

2.7.4 Efficacy of Collaborative Filtering

As a spam filtering solution Collaborative Filtering faces many challenges that can significantly affect its accuracy and precision in filtering spam messages. The concept of distributive filtering can be counter-productive if the various elements are not adequately synchronised. For Collaborative Filtering to be effective is must ensure that it meets the challenges of performance, scalability and trust. The efficacy of a Collaborative Filtering solution is affected by the number of users that participate in generating the spam rule. Once this number increases into the millions, targeting and interconnecting them can be a challenge with unpredictable results. Collaborative Filtering algorithms suffer immensely with scalability problems predominantly when computational resources exceed acceptable levels. Ultimately, the trust scheme employed by the Collaborative Filtering solution is crucial to its success. Hackers will attempt to destabilise the system by providing false information to the spam rule set (Kong, et al., 2006; Su and Khoshgoftaar, 2009).

Since the establishment of Collaborative Filtering and in particular Blacklists, spammers and hackers have been working together to develop alternative approaches to spamming including the implementation of botnets (Alkahtani, Gardner-Stephen and Goodwin, 2019; Bhowmick and Hazarika, 2016). A 'Spam-bot' refers to a compromised computer on the internet. Thus, the term 'botnet' refers to a network of compromised machines controlled by a 'botmaster'. When compromised and controlled a botnet can be implemented to deploy a series of attacks including denial of service attacks, malware dissemination, phishing and spamming. Consequently, the slightest vulnerability within a collaborative network can result in a compromised spam rule set (Bhowmick and Hazarika, 2016).

A major challenge that impacts the efficacy of a Collaborative Filtering solution is data sparsity. To establish a true opinion on an entity the user-matrix must be sizeable enough. If inadequate, the predications and recommendations of the Collaborative Filtering solution will be inaccurate and can be challenged. The data sparsity problem appears under several conditions such as the introduction of a new item, email address or IP address. The new entity cannot be recommended until a prescribed percentage of users have rated it. If the problem persists, the Collaborative Filtering solution will make a prediction based on previous trends as in the case of memory-based Collaborative Filtering. This subsequently diminishes the

efficacy of the Collaborative Filtering solution and can expose it to attacks (Su and Khoshgoftaar, 2009).

One such threat is a 'shilling' or 'profile injection' attack. In a shilling attack, attackers' forge rating profiles and inject them into the rating matrix of recommender systems such as collaborative spam filters. Spammers are motivated by profits and continuously discover innovative ways to target and exploit a system to realise their goals. Consequently, to achieve maximum benefits spammers are prone to deploy a shilling attack using a group of users or bots to rapidly inject biased opinions on an item either to drive its recommendation ratings up (known as a push attack) or to reduce its recommendation ratings (regarded as a nuke attack) (Vyas, Prajapati and Gadhwal, 2010; Zhou, et al., 2018).

Whilst there are several studies conducted and solutions proposed regarding shilling attacks, it remains a major problem that continues to plague recommender systems and in particular Collaborative Filtering. Shilling attacks tend to evolve with variations to the shilling detection methods. When attackers are cognizant of a shilling detection mechanism, they promptly react to diminish its effectiveness, consequently reducing the efficacy of the system (Fan, 2010; Zhou, et al., 2018).

This section has discussed the two commonly adopted approaches to spam filtering. Furthermore, it has also exposed the various vulnerabilities associated with each approach. The ability to destabilise and reduce the efficacy of filtering systems, such as Collaborative and Content-based filtering, affirms the motivation for the implementation of spam email as an ideal carrier for covert communications. The next section presents and discusses the implementation of spam email to perpetuate crime.

2.8 Crimes Related to Spam Email

This section discusses a selection of cases where attackers or co-conspirators employ spam to perpetrate crimes. Additionally, it highlights the vulnerability of emailing and filtering systems, and the subsequent exploitation attributed to its presumed innocuity.

In a journal article, Yu (2015), addresses five cases associated with email forensics. In one such case, he discusses the murder of a housewife. In this particular case the housewife was shot dead in her home. The murder scene showed no traces of forced entry into her home, and the murderer left no physical evidence for investigators to work with. It was, therefore, quite apparent to the investigators that the woman was murdered by someone close to her. Naturally, the husband was the prime suspect. The investigation, however, was limited as the husband had an alibi. Despite the alibi, investigators were convinced that the husband committed the murder or hired someone else to do so on his behalf. Investigators then started to pursue other opportunities to unearth any evidence that might assist in the case.

After much effort and with no leads, the investigation was finally handed over to the cyber-crimes unit to see if they could unearth any new evidence. Cyber forensic investigators started by examining the husband's hard drives and electronic communications. Unfortunately, after intensive analysis using keyword searches and common filtering techniques on all his email accounts it yielded no incriminating evidence or anything that could link him to the murder. Investigators did not know at the time that they made a mistake when they decided to habitually dismiss messages in the spam folders as they were presumed innocuous. After travail efforts on normal emails, investigators finally resorted to analysing the husband's spam folders (Yu, 2015).

Upon investigation, investigators discovered that many of the spam emails were legitimate as they carried similar traits? associated with normal spam email. Only after several attempts and a thorough analysis of the spam emails did investigators find a clue. A hidden message in an odd-looking spam email that had no URL links embedded in the narrative and nor was it sent out in bulk to other inboxes, two common traits associated with spam emails (Yu, 2015).

Implementing several combined techniques investigators soon discovered a hidden message that read 'first Thursday next month in the Walmart on west 9am isle 1'. This was the meeting schedule communicated between the murderer and the suspected husband. Investigators later discovered that the spam email in question was created using a well-known website spammimic.com. This website uses probabilistic context-free grammar (PCFG) to conceal a secret message into a body of text that mimics a typical spam email (Bajaj and Pieprzyk, 2014;

Yu, 2015). The husband was consequently arrested and convicted based on these findings. So why was it difficult to initially detect the hidden message?

It is the mere perception to the innocuity of spam email that tunnel-visioned investigators into discounting spam as a potential source of evidence to the investigation. According to Yu (2015, p73), *"To our best knowledge, there are no existing digital forensics tools that can detect such encryption".*

It is this research's objective to highlight the hidden threat posed by the assumption that spam email is innocuous. To demonstrate this Yu (2015) discusses another case whereby a criminal wanted to encrypt an email using some freeware available on the internet, but as he did not want the encrypted email to be noticeable, he elected spam email as a cover. In this case the criminal simply used an existing spam mail and merely inserted his encrypted message into it. This served as two layers of protection, firstly the email appears to be spam and due to the assumption that spam is innocuous it did not receive much attention. More so the criminal in question sent the message in bulk to a group of randomly harvested emails (Yu, 2015).

In some cases, criminals would attempt to manually encrypt an email message using a computer-aided encryption tool. To illustrate this Yu (2015), presents a case of terrorism investigation for an alias known as 'Dennis Cook'. Cook was suspected to be involved with a White supremacy group that were plotting various terrorist acts. When discovered, Cook tried to erase all digital evidence but failed to do so. It was eventually discovered that Cook and his accomplices were covertly communicating via carefully-crafted spam emails regarding trivial topics like the weather or TV shows. This source of potential evidence was initially disregarded as innocuous by the investigators' tunnel vision of spam which resulted in them disregarding it (Yu, 2015).

There are several cases which demonstrate that the presumed innocuity and persistence of spam predestines it to persist as a dominant vessel for cyber-crime (Clark, et al., 2018; Yu, 2015). Other threats advocated by the implementation of spam include Distributed Denial of Service attacks, Social Engineering, BEC and Malware injections. Considering that email is a push, non-repudiative technology spammers have employed innovative techniques to

distribute spam and evade detection or prosecution (Brush and Ferguson, 2019). This is therefore similar to the implementation and control of botnets discussed in the previous section.

The *Spamhaus Botnet Threat Report 2019*[11] has revealed that there has been a sustained upward trend in the number of new botnets detected since 2017. According to this report, significant increases were also observed in the number of Malware associated with the botnet Command & Control (C&C) Servers - a preference for cyber-criminals when deploying crimeware kit. Consequently, since 2017 the number of newly detected botnet C&Cs has doubled from 9 500 to 17 602 resulting in significant increases in malware.

2.9 Chapter Conclusion

It has been the goal of this chapter to identify, analyse and disclose aspects of the emailing system to address the next research objective (OB 1):

Identify and determine the vulnerabilities and motivation for the election of spam email as a covert channel.

This chapter discusses the intrinsic lack of security of the SMTP transport protocol, the vulnerabilities and several challenges associated with content and collaborative-based spam filtering approaches, the perpetration of cyber-crime through the implementation of spam, the apathy of users and cyber-professionals through the assumption of the innocuity of spam, and the apparent lack of security policy enforcement.

These factors adequately substantiate why several reports and studies reveal that spam email remain ubiquitous and an ideal medium for its deployment or other email-related attacks (Clark, et al., 2018). Apart from the apparent security challenges, the mere perception of the innocuity of spam email offers ample motivation for the election of spam as a covert channel. Consequently, one of the primary requirements for covert communications would be to remain imperceptible and spam email clearly fits the bill which, in turn, achieves the objective of establishing spam email as an ideal carrier for covert communications.

[11] *https://www.spamhaus.org/news/article/793/spamhaus-botnet-threat-report-2019*

Increased probability of false positives, irregular precision and accuracy levels attribute to vicissitudes in the training sets, poorly defined and outdated spam rules and other forms of attacks, which all contribute to dissolving the efficacy of both content- or collaborative-based spam filtering solutions. These, together with the other factors presented and examined in this chapter address the research objective mentioned and by inference the secondary research question 1 (SQ1):

Why is spam email considered an ideal carrier for covert communications?

The next chapter presents and discusses the different forms of steganographic approaches in particular linguistic steganographic.

3

Steganography

This chapter presents a discussion on the various concepts and modalities of steganography. Additionally, it provides a detailed discussion on Linguistic Steganography to establish the rationale for its adoption within this research study and finally discusses the different linguistic steganalysis[12] approaches in order to address the Research Objective 2 (OB2) – ***To identify existing linguistic steganalysis methods and determine its suitability for the detection of stegospam files generated using a probabilistic context-free grammar (PCFG) from non-stegospam files.***

Introduction

A number of spam concepts and factors including its affordability, convenience, pervasiveness and presumed innocuity were presented and discussed in Chapter 2, which establishes the rationale and motivation for the election of spam email as an ideal cover medium for covert-based communications. Consequently, this chapter examines the different steganographic concepts and approaches implemented to conceal one message within another, by a keen focus on linguistic steganography and its implementation through the use of spam email. Before the key sections of this chapter are examined, an introduction to steganography is necessary.

Agarwal (2013), defines steganography as the art and science of concealing a message inside another message without arousing any suspicion. The word steganography originates from the Greek words '*steganos*' meaning '*hide*' and '*graphein*' meaning 'to write' (Roy and Manasmita, 2011; Chaudhary, Dave and Sanghi, 2016). The concept of concealment was first established in ancient times, where messages were concealed by writing them on the stomach of rabbits, or tattooed on the scalps of slaves. As communication methods improved,

[12] *Steganalysis refers to the discovery of the existence of hidden information (Richer, 2019).*

so too did the art of concealment. The proliferation of the internet and mobile computing has promoted more sophisticated and innovative techniques of steganography across heterogeneous devices. In steganography the message to be concealed is termed the 'secret message' which is encoded using a key, more specifically termed the 'stego-key', into a piece of random medium such as a plaintext,[13] picture, audio or video file referred to as the cover medium. The stego-object (secret message and cover medium) can only then be detected and decoded by the intended recipient holding the stego-key. Steganography presents a seemingly innocent approach to covert communications even in the presence of an observer. Text-based steganography, in general, uses a natural language (i.e. a human language like English) as a cover since the language itself provides the security against any arbitrator that accepts natural language as a communication medium (Huanhuan, et al., 2017).

Unlike data encryption which implements cryptographic techniques to encrypt sensitive information, steganography camouflages sensitive information by obscuring it from view. A number of different techniques to steganography have been introduced but are subjected to the cover medium chosen i.e. a text, audio, image or video file (Changder and Majumder, 2013). The amount of information that can be encoded into the cover medium is dependent on the amount of redundant information the cover medium possesses. Consequently, audio, image, or video are preferred cover mediums due to their high volume of data redundancy when compared to text which offer very little or no redundant information (Agarwal, 2013; Changder and Majumder, 2013).

When electing a cover medium, steganographers have to consider several pertinent properties which include the degree of imperceptibilty – the ability to remain undetected, the degree of robustness – the ability to resist transformation attempts and the capacity level of the cover medium (Abdul-mahdi, Yahya and Ahmad, 2013). These and other relevant aspects are discussed in greater detail in the following sections.

13 In this research study the use of the word "plaintext" or "text" in reference to a text-based cover medium is used interchangeably.

The structure of this chapter follows the outline below**:**

- **Section 3.1** - Definition and Background of Steganography
- **Section 3.2** - The Steganographic Process
- **Section 3.3** – Properties to consider when electing a steganographic approach
- **Section 3.4** - Steganography Vs Cryptography
- **Section 3.5** – Classification of Steganography based on the cover medium
- **Section 3.6** – Text-Based Steganography
- **Section 3.7** – Linguistic Steganographic Encoding Algorithms
- **Section 3.8** – Linguistic Steganalysis
- **Section 3.9** – Review – Linguistic Steganography

3.1 Definition and Background of Steganography

This section provides a detailed discussion on the definition and background of steganography by building on the introduction with the intention of establishing a sound understanding of key steganographic concepts and methodologies. Understanding the role and scope of steganography within the cyber-security sector is vital to establish its purpose within this research study.

The proliferation of the internet has seen the number of users and organisations grow exponentially. However, transmitting information across the internet has become increasingly risky and highly susceptible to eavesdropping, man-in-the-middle, denial-of-service and phishing attacks. Subsequently, various approaches have been adopted by users and organisations to secure their information in transit such as virtual private networks, cryptography and steganography (Sharma, Rekha and Manisha 2016).

Even though dedicated private networks are effective, they are extremely expensive and have a constrained access. Cryptography, in contrast, is not constrained to a physical point of access and is inexpensive to implement. Cryptography refers to the science of data encryption, which employs encryption algorithms and techniques to make a message incomprehensible to an unauthorised receiver; but it does not conceal the existence of the encrypted message. The encrypted message can only be decrypted using a decryption key. A

common drawback of data encryption is that there are numerous methods and tools that can be implemented to decrypt the encrypted message once it has been detected (Bhat, et al., 2017; Changder and Majumder, 2013; Richer, 2019).

Steganography offers a different approach to the problem of information security. As defined in the introduction, steganography refers to the art and science of concealing sensitive information in an innocuous cover medium such as an image, audio, video or text file. Only the sender and the intended recipient has knowledge of the presence of the hidden message. This level of obscurity implies that the appearance of the cover medium is manipulated in such a manner that the hidden message is imperceptible by any intermediary, consequently adding an additional layer of protection against detection (Chaudhary, Dave and Sanghi, 2016; Singh, Diwakar and Upadhyaya, 2014).

One of the earliest forms of digital steganography is watermarking which was developed for the protection of intellectual property i.e. copyrighting. Digital watermarking allows an author or publisher to watermark a digital book or document to restrict the illegal reproduction of it. Digital watermarking has since been extended to audio, video and software products and serves as a suitable method for identifying the owner, distributor, or authorised user of a digital product (Shih, 2017).

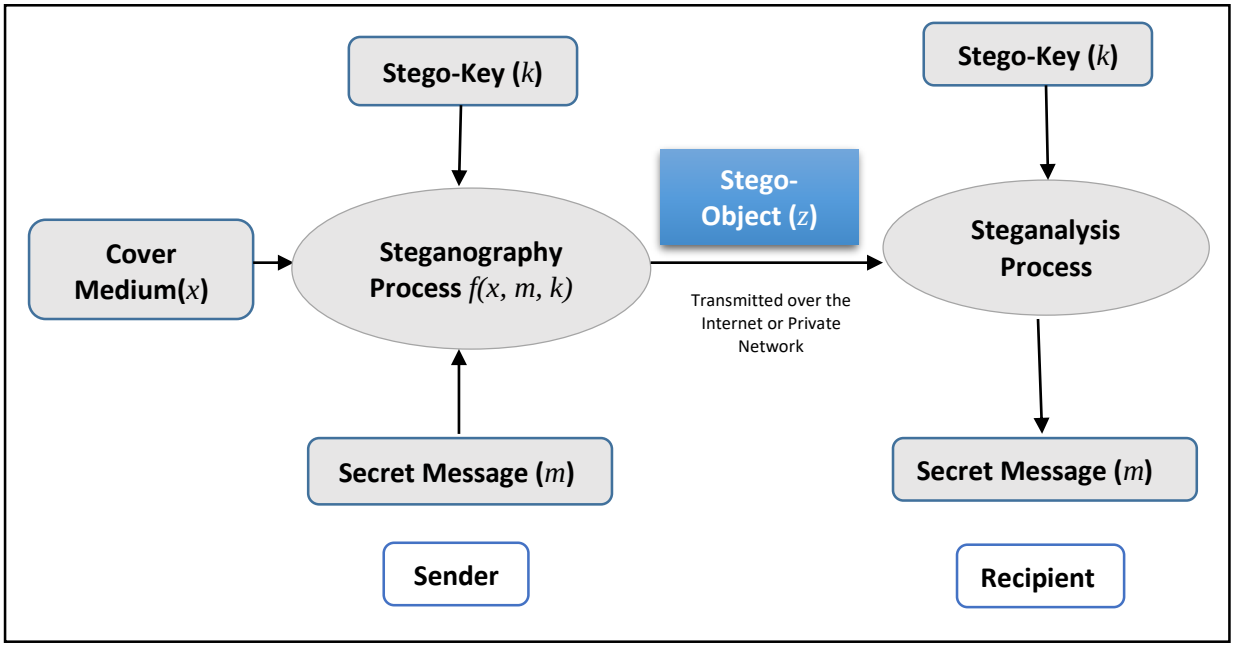
Digital watermarking and other steganographic concepts are discussed later in this chapter, for now there are a few fundamental terms and components need to be taken cognizance of, these include:

- **Cover medium.** This is fundamentally a carrier or a file that it used to conceal a message.
- **Stego-key.** This refers to an additional piece of information that is used to further encode a message.
- **Stego-object** or **Stego-medium.** This refers to the end product of steganography that encapsulates the cover medium and the hidden message.
- **Stego-text.** This refers to a stego-object whose cover medium is purely a text body that has undergone some form of steganographic process.

- **Steganalysis.** This is the process or algorithm that involves the discovery of the existence of concealed information in a cover medium (Richer, 2019).
- **Secret message.** This refers to the information that needs to be hidden or concealed within a cover medium.
- **Survivability**. This means that all data processing takes place between the sender and receiver and does not destroy the secret message. Additionally, the transmitted information must be extractable and comprehensible (Almohammad, 2010).

3.2 The Steganographic Process

In the preceding section a few fundamental concepts were identified; however, in this section the different steps involved in the steganographic process are considered to ascertain when and how a secret message is concealed within a cover medium. Understanding this process is vital to this paper in terms of recommending an alternative approach to the detection of conceal messages in plaintext cover mediums such as a spam email (particularly since the process in principle is similar for all steganographic approaches).



Source: Sadi (2015).
Figure 3.1 - The Steganography process

Figure 3.1 above demonstrates the process of embedding a secret message (m) into a cover medium (x) via a steganographic process ($f(x, m, k) \mapsto z$) using a stego-key (k) to create the

stego-object (z). The stego-object (z) is then transmitted over a public domain such as the internet or through a private network to the intended destination. The recipient must have the stego-key (k) that was used in steganographic process ($f(x,m,k)$) in order to extract the secret message (m) from the stego-object (z) during the steganalysis process. During the steganography process, the sender must ensure that the stego-object appears identical to the original cover medium in order to avoid suspicion (Sadi, 2015). It is important to acknowledge that the process illustrated in Figure 3.1 is independent of any cover-medium type or additional security that may be added by the originator such as data encryption or encoding schemes. At this juncture it would not be possible to evaluate the process on the properties of imperceptibility, robustness and capacity as no cover medium is defined. However, an understanding of these properties is essential in evaluating the cover medium and its associated embedding scheme.

3.3 Consideration of Properties for a Steganographic Approach

This section discusses the key properties that are considered when electing a steganographic approach which include imperceptibility, robustness and capacity as stated in the introduction. A sound understanding of these properties is essential when assessing different cover mediums and their embedding techniques.

In steganography it is not possible to maximise all these properties simultaneously. Consequently, a satisfactory balance must be achieved by the steganographic approach elected. When steganography is implemented as a method for covert-based communications, it is typical that imperceptibility becomes the foremost requirement, whilst robustness and capacity are to a certain extent sacrificed. This is largely ascribed to the underlying concept of ensuring that the stego-object remains undetected (Abdul-mahdi, Yahya and Ahmad, 2013). The following paragraphs discuss these properties in greater detail.

3.3.1 Imperceptibility

Imperceptibility refers to the ability of a cover medium to efficaciously conceal a secret message to avoid detection by an observer or a detection system. If the cover medium has a high level of imperceptibility it implies that the data embedded in it has a high degree of

survivability. It is important to note that the level of imperceptibility is subject to the type of cover medium and the associated encoding or transformation methods implemented (to be discussed in section 3.4).

Additionally, the hidden message must not only be invisible perceptually, but statistically as well to evade detection. Hence, a steganographic approach is deemed secure if the cover medium and the stego-object are statistically equivalent. To achieve this, the characteristics and attributes of a cover medium should not be distorted to the extent that it affects the statistical balance between the cover medium and the stego-object. Consequently, the detection of any statistical anomalies may be used to prove the existence of a hidden message (Almohammad, 2010). The forms of statistical analysis adopted by different steganalysis are discussed in greater detail in section 3.6 of this chapter.

In summary, the security of a steganographic approach is defined in terms of imperceptibility, which is assured when a steganalysis cannot statistically differentiate between the cover medium and the stego-object (Taouil, et al., 2016).

3.3.2 Robustness

The next property to consider is robustness. Robustness refers to the ability of the concealed message to remain unaffected even if the stego-object is subjected to threats that could destroy or modify it intentionally or unintentionally. A robust steganographic algorithm makes a stego-object extremely difficult to modify or destroy. Moreover, it defines the survivability or resiliency of the secret message against such threats (Almohammad, 2010; Ahvanooey, et al., 2019).

Most steganographic approaches use both private and public networks as their communication mediums, which do not affect the integrity of the stego-object. Therefore, they do not considered robustness of the stego-object a priority as the recipient receives exactly what the sender transmitted. Consequently, this exposes the stego-object to different forms of attacks including decomposition, file format conversion and digital-to-analogue conversion. To be considered a favourable option, therefore, a steganographic algorithm must consider the robustness of the stego-object it generates (Almohammad, 2010; Ahvanooey, et al., 2019).

Steganographic tools such as SNOW and STEG implement a multi-layered architecture to improve the robustness of the stego-object. This entails the implementation of cryptography to encrypt the secret message and then conceal it using steganography. Therefore, if the secret message is detected, the encryption helps to minimise the resulting damage, since the data is not exposed, only the knowledge that a secret message was transmitted (Ahvanooey, et al., 2019).

However, steganalysts (that is, a person who performs steganalysis techniques on a suspected stego-object) can simply destroy the suspected stego-object if he/she cannot determine the contents of the file as a precautionary measure. Therefore, this once again affirms the need for a stego-object to remain as imperceptible as possible (Ahvanooey, et al., 2019). Consequently, the assumed innocuity and ubiquitous nature of spam email fits this mold, affirming its election as a cover medium and focus in this research study.

3.3.3 Capacity

Steganographic capacity refers to the maximum number of bits that can be embedded within a cover medium with a negligible probability of detection by a steganalysis. Hence, the embedding capacity is likely to be larger than the steganographic capacity. Furthermore, the size of the secret message relative to the size of the cover medium is known as its embedding rate (Almohammad, 2010). Steganographic approaches implemented for covert communications aim to capitalise on the steganographic capacity to minimise the perception of a hidden message within a stego-object (Ahvanooey, et al., 2019; Wang & Wang, 2004). Steganographic capacity is subjected to the cover medium implemented. Image, audio or video files carry a lot of redundant data which can be used to conceal much larger messages and, therefore, have a higher embedding rate. Text-based steganography, on the other hand, is a lot more challenging as plaintext files offer very little to no redundant data. A digital text file is written, saved and retrieved by a computer in precisely the same way it appears to the naked-eye (Roy and Manasmita, 2011). Therefore, further innovative techniques have been proposed and implemented to embed a secret message within plaintext cover mediums. These techniques are discussed later in this chapter.

The consideration of these properties are essential when addressing the second sub-research question (SQ2):

> **Which linguistic steganographic techniques are favoured in order to embed a secret message within plaintext files?**

Therefore, a deeper analysis of these properties based on the different cover types and corresponding steganalysis techniques are discussed in Section 3.3 of this chapter.

3.4 Steganography vs. Cryptography

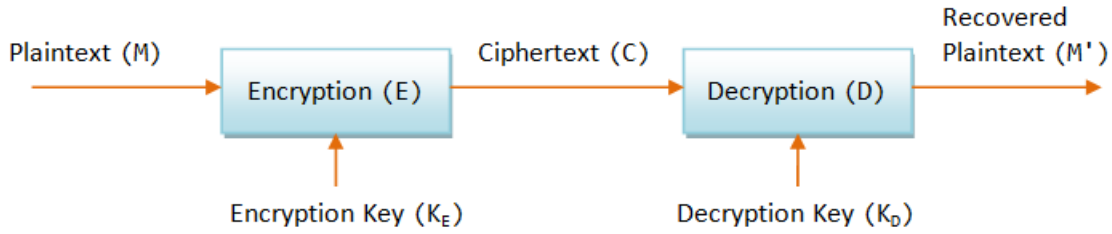
This section considers the differences between steganography and cryptography. Understanding the differences between these security methods will provide a better level of understanding and help identify the motivation for the implementation of steganography. To ascertain when steganography is the preferred method, the characteristics and application of each approach needs consideration and then its advantages and disadvantages need to be deliberated.

The security of information has always been an essential requirement for users dating far back as 1900 B.C., before the days of computing technology (Bhat, et al., 2017). Over the following decades, however, cryptography and steganography emerged as the two most preferred methodologies implemented by cyber-security practitioners to secure data communications across private and public networks. The exponential growth of digital communications due to faster, ubiquitous internet-enabled devices and services has increased the demand for more robust and reliable security. Therefore, in many cases cryptography and steganography have been integrated into a hybrid, multi-tiered security solution, where the data is initially encrypted and then embedded into an innocuous cover medium (Bhat, et al., 2017; Koluguri, Gouse and Reddy, 2014; Sharma, Rekha and Manisha, 2016).

According to Raphael & Sundaram (2011, p627), cryptography is defined as *"the art and science of transforming data into a sequence of bits that appear random and meaningless to a side observer or attacker".* As previously mentioned, cryptography was implemented long before computers were even invented. In fact, one of the earliest implementations of cryptography dates back to the age of Julius Caesar. In Caesar's time a basic mono-alphabetic

system (i.e. a system that uses a single alphabetic set for substitution) of encryption was introduced in which each character was replaced by the third character in succession, now aptly named the 'Caesar Cipher'. Today, several encryption algorithms are in use including Data Encryption Standard (DES), Triple DES (3DES) and Advanced Encryption Standard (AES) (Edgar and Manz, 2017; Raphael and Sundaram, 2011;).

To analyse a cryptographic system a process known as cryptanalysis is employed to discover any vulnerabilities or information leakage in the system (Edgar and Manz, 2017). During the cryptographic process (illustrated in Figure 3.2), the information to be secured is encrypted using an encryption key *(K_E)*. However, the drawback to this approach is that the existence of the encrypted data (ciphertext) is known, hence exposing it to brute-force and other forms of attacks (Edgar & Manz, 2017; Raphael and Sundaram, 2011;).



(Source: Chuan, 2009)
Figure 3.2 – The symmetric encryption and decryption process

Figure 3.2 above illustrates the process of encryption and decryption, which can be expressed using the following mappings:

$$E(M, K_E) \mapsto C$$

$$D(C, K_D) \mapsto M'$$

Ideally, M′ = M

In symmetric cryptography both the encryption and decryption keys are identical, whilst in asymmetric cryptography these keys are different. Steganography, on the other hand, does not use encryption keys, rather it implements what is known as 'null ciphers' to conceal a secret message. Null ciphers are unencrypted messages embedded in a cover medium. Concealed messages are difficult to detect particularly if they are concealed within an innocuous and ubiquitous medium such as spam email (Edgar and Manz, 2017; Raphael and Sundaram, 2011).

In cryptography the strength of the encryption algorithm determines the survivability and robustness of the cryptographic method, whereas in steganography it is the embedding and transformation methods and the cover medium. The application of cryptography or steganography is subject to the intent of the applicator. Taking into consideration that cryptography encrypts a message and does not conceal it (hence the mere presence of the ciphertext arouses suspicion) which, in fact, defeats the objective of covert communications. However, steganography can be useful when the use of cryptography is limited or prohibited. Here steganography can circumvent such restrictions to communicate covertly. Conversely, steganography fails if an observer discovers the existence of a hidden message within a cover medium (Cole, 2003; Edgar and Manz, 2017; Raphael and Sundaram, 2011).

The question then is: Which approach is better? There are a few factors to consider. Both methods are used to secure sensitive information, especially when one wants to transmit such information across a public network such as the internet. If cryptography and steganography are analysed in terms of bandwidth efficiency, cryptography would be the preferred method. This is attributed to the fact that cryptography does not really add much of a payload to a plaintext file in order to encrypt it. Steganography on the other hand, embeds a secret message within the cover medium adding to its overhead, thus consuming more network bandwidth than cryptography. The downside to this is that the ciphertext is not concealed and can be easily detected. Therefore, if innocuity is the *modus operandi*, then steganography would undeniably be the preferred method. Evidence of this is seen in the adoption of steganography by companies that operate e-Commerce stores to covertly track online user activities via URL embedding, hidden fields, or cookies. In terms of detection and survivability both systems will survive unless the cipher-key is obtained in cryptography or the stego-key in steganography, hence exposing the data (Cole, 2003; Raphael and Sundaram, 2011).

Table 3.1 below provides a summary of the differences between cryptography and Steganography.

Item No	Cryptography	Steganography
1.	It is a technique used to convert the secret message into an incomprehensible form	It is a technique used to hide the existence of the communication
2.	Alters the overall structure of the data	Does not alter the structure of the cover medium.
3.	The final result obtained is termed the ciphertext	The final result obtained is termed the Stego-Object
4.	Once it has discovered , no one can easily read the data	Once it has been discovered anyone can read the data *(unless first encrypted)*

Table 3.1 – Differences between Cryptography and Steganography

In this section the application and differences between cryptography and steganography were discussed. It is imperative to have a fair understanding of these two approaches to data security as a few linguistic steganographic approaches employ both approaches to add an additional layer of security (as further discussed in Section 3.6 of this chapter). The next section provides a discussion on the classification of the different forms of steganography based on the cover medium elected.

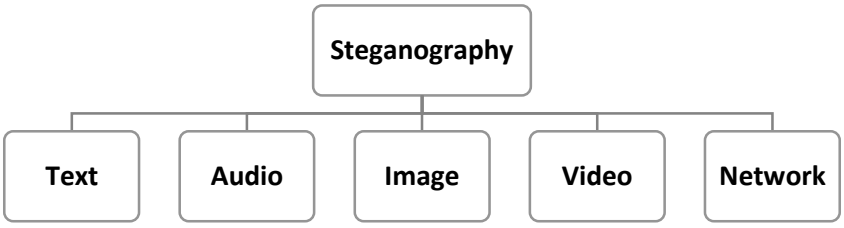
3.5 Classification of Steganography based on the Cover Medium

This section provides a brief overview of the different modalities of steganography to gain a better understanding of each form, particularly linguistic steganography, which is the focus of this research study.

Most all digital file formats with a high degree of redundancy are commonly used in steganography. Redundancy refers to the volume of secret data that is capable of modification without the possibility of detection. Modern steganography is categorised based upon two main schemes for the taxonomy of steganography. The first scheme is based on the file or cover type and the second is based on an embedding technique (Hamid, et al., 2012). As stated in the previous sections the embedding technique (i.e. the technique used to conceal sensitive information) is subject to the cover medium chosen. Consequently, this section provides an overview based on the former scheme. This helps in understanding the

limitations and motivation for the implementation of a cover medium based on the properties of robustness, capacity, and the imperceptibility of the stego-object.

Given the research scope mentioned earlier in this chapter, greater emphasis is placed on text-based cover mediums to support our understanding of the different embedding techniques used to conceal a secret message, the motivation for its implementation and vulnerabilities when compared to other cover mediums. Figure 3.3 below illustrates the classification of the different steganographic approaches based on the cover medium adopted.



(Source: Koluguri, Gouse and Reddy, 2014)
Figure 3.3 – Steganography types classified by digital carrier types.

3.5.1 Audio-based steganography

In the past decade, numerous techniques for data concealment in audio sequences have been presented. All the developed techniques take benefit of the perceptual properties of the Human Auditory System (Kiah, et al., 2011). In this approach sensitive information is concealed using an audio file as a cover medium. To ensure survivability and imperceptibility the host audio file must have the same characteristics before and after the embedding process. Common audio file types include MP3, WAV and MPEG. Presently, audio-based steganography is frequently used for sound watermarking to conceal the information of an artist or author. Audio-based steganography implements numerous embedding techniques such as Least Significant Bit (LSB) coding, Parity coding, Phase coding, Spread Spectrum and Echo Hiding. These embedding techniques allow audio-based steganography to achieve a higher payload capacity with lower imperceptibility rates when compared to text-based steganography (Chaudhary, Dave and Sanghi, 2016; Hamdan and Hamarsheh, 2017).

3.5.2 Image-based Steganography

Digital images are considered the most common cover medium due to its popularity on the internet and high capacity of redundant data. An image is defined as a pattern of pixels[14] that control the colour and light intensities of an image. When selecting an image as a cover medium there are a few important factors that must be considered which directly influence its payload capacity and robustness. One such factor is the image quality, fundamentally an image with a higher quality has a higher density of pixels and subsequently presents a higher payload capacity as opposed to an image of lower quality. Another critical factor is the encoding format implemented, since different encoding formats can withstand different types of image manipulation such as cropping, compression and editing, which can affect the robustness of the cover medium, hence destroying or distorting the embedded message. There are numerous embedding techniques that can be used to conceal a secret message in an image file, all of which have corresponding strengths and vulnerabilities (Cole, 2003; Hamid, et al., 2012). In their paper Hamid et al. (2012), present a review and analysis of the different embedding techniques used on an image cover medium, namely LSB, Transform Domain, Spread Spectrum, Statistical Techniques, Distortion Techniques, and File and Pallet Embedding that directly influence its capacity, robustness and imperceptibility. The impact of these embedding techniques is tabulated below in Table 3.2.

	LSB	Transform Domain	Spread Spectrum	Statistical Techniques	Distortion Techniques	File and Pallet Embedding
Imperceptibility	High	High	High	Medium	Low	High
Robustness	Low	High	Medium	Low	Low	Low
Payload Capacity	High	Low	High	Low	Low	High

(*Source:* Hamid, et al., 2012).
Table 3.2 – Review of image file embedding techniques

3.5.3 Video-based steganography

Video files are essentially a collection of images and sounds, therefore, most of the embedding techniques applied on images and audio covers can also be applied to video files. One of the obvious benefits of video files is that it supports a large payload capacity for the concealment of data. Consequently, video-based steganography is fundamentally a

[14] *"pixel" is an abbreviation for "Picture Element", which are combination of little dots that make up the images on a computer display* (Christensson, 2006).

combination of image and audio-based steganography (Hamid, et al., 2012). One of the embedding techniques used in video-based steganography involves the exploitation of the YCbCr colour spaces which is used to develop the images in video and digital photography systems:

> *(YCbCr represents colours as a combination of three values: Y – the luminosity (brightness), Cb – the chrominance (colour) of the blue primary and Cr – the chrominance of the red primary)* (Elsadig, et al., 2009).

3.5.4 Network steganography

The fundamental objective of steganography is to enable covert-based communication between entities across a private or public network. Concealing information within a cover medium is beneficial; however, one of the challenges of steganography is to guarantee that the cover medium reaches its destination. In this regard, several approaches have been used to securely transmit a secret message across a network. These include hiding information within a cover medium to create the stego-object, then distributing the stego-object using some form of network transmission such as an email system, an FTP (File Transfer Protocol) application, or by posting the stego-object on a web site. Another approach would be to conceal sensitive information in network packet headers such as the Internet Protocol (IP) header. Protocol packet headers contain many fields that are optional or unused which are ideal for hiding information. Finally, the assumption of an innocuous overt protocol such as HTML, camouflaging the secret message to appear as normal web traffic allows the stego-object to penetrate firewalls and Intrusion Detection Systems (IDS) (Elsadig, et al., 2009).

As previously stated, the focus of this research study is centred on text-based steganography, in particular, linguistic steganography. Therefore, only a brief discussion on other forms of steganographic is presented. The next section provides a comprehensive discussion on text-based steganography not discussed in this section.

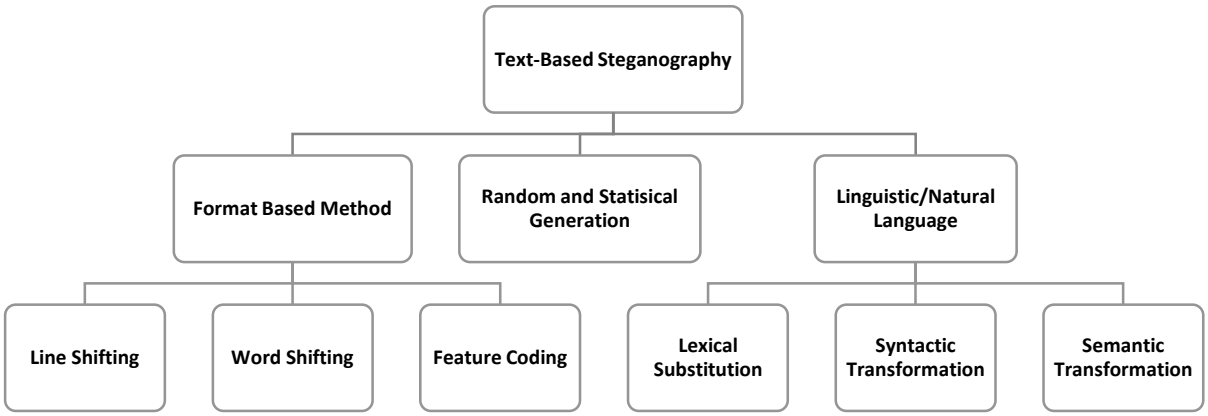
3.6 Text-based Steganography

In this section text-based steganography is considered and examined − a key focus area of this research study. Furthermore, this section assists in addressing the secondary research question SQ2:

Which linguistic steganographic techniques are favoured in order to embed a secret message within plaintext files?

Text-based steganography refers to the art and science of concealing a secret message within another body of text such that it is imperceptible to an observer. Additionally, text-based steganography remains one of the most challenging of all steganographic types due to the lack of redundant data within the text cover medium (Bhat, et al., 2017).

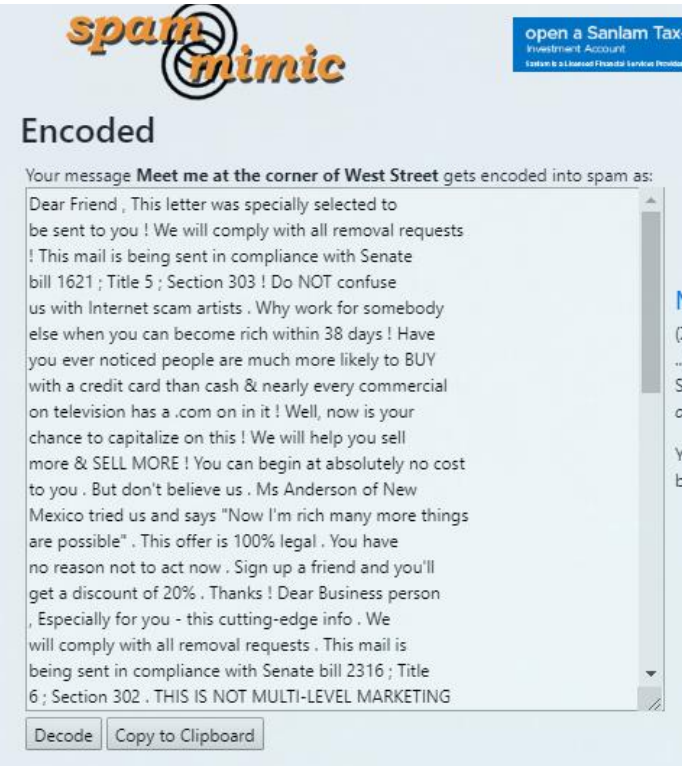
Embedding a secret message within a corpus of text can range from simple text formatting to lexical modification (Ahvanooey, et al., 2019). Based on the concepts of text generation, transformation and encoding techniques, text-based steganography can be categorised into Random and Statistical Generation, Format-Based and Linguistic steganography as illustrated in Figure 3.4 (Koluguri, Gouse and Reddy, 2014). Additionally, this diagram and Section 3.5 will provide further insight when proposing a steganalysis for the detection of stegospam files generated using linguistic steganographic techniques.



Source: (Koluguri, Gouse and Reddy, 2014)
Figure 3.4 – Text-based Steganography

3.6.1 Random and statistical generation

To evade visual or comparative analysis through the use of the original cover text, steganographers commonly choose to generate their own cover texts through various text transformation techniques (as discussed later in this section). One such technique involves the implementation of a Context-Free Probabilistic Grammar (CFPG*)* such as Spammimic, NICETEXT or TEXTO that obeys the statistical properties of word length and frequency to generate syntactically and statistically accurate cover text that is as close as possible to a natural language (Koluguri, Gouse and Reddy, 2014; Zhi-li, et al., 2008). The purpose of this approach is to simultaneously embed a secret message into the generated cover text such that it complies with the statistical and grammatical rules of the language to remain seemingly innocuous. As mentioned, Spammimic is a CFPG that is freely available on the Internet. Spammimic allows a user to automatically generate a text cover that mimics an innocuous spam email through the use of linguistic steganographic techniques. Figure 3.5 below illustrates the cover text generated by Spammimic for the secret message '*Meet me at the corner of West Street'.*



Source: http://www.spammimic.com

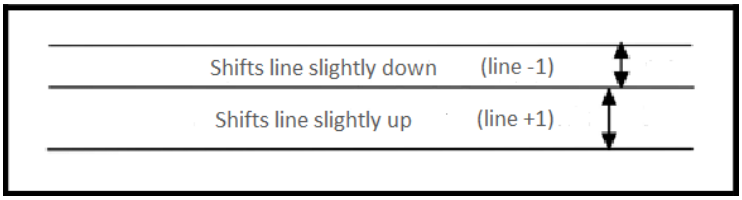
Figure 3.5- Random and statistical text generation using Spammimic.

3.6.2 Format-based steganography

The embedding techniques employed in format-based steganography are to conceal a secret message by marginally adjusting the appearance of the text cover so that it is imperceptible to the human eye. There are numerous approaches to format-based steganography that have been proposed and implemented over the past few decades. The more common approaches include: Word Shifting, Feature Coding, Inter-Sentence Spacing, Line Spacing and Inter Word Spacing (Chaudhary, et al., 2016). As previously stated, the payload capacity of a text-based cover medium is extremely low as it possesses very little redundancy in order to conceal a secret message. Consequently, steganographers had to develop innovative techniques to embed the secret message within a text cover (Changder and Majumder, 2013; Chaudhary, Dave and Sanghi, 2016).

As specified, this research study is focused on the detection or classification of stegospam files generated using linguistic steganography; however, a few of the format-based approaches mentioned are discussed next to provide a holistic overview of three different embedding techniques employed in text-based steganography:

- **Line Shifting Method.** This method embeds sensitive information into a cover medium by shifting the lines of text vertically to some degree. A code-word assigned to a document specifies which lines of text will be shifted. There may be a '*+1*' for a line shifted up and a '*-1*' for a line shifted down as illustrated in Figure 3.6. This technique uses the differential encoding technique for achieving performance and robustness. However, line shifting can be exposed using special tools that are designed to evaluate line distances. Once discovered, necessary changes can be made to destroy the hidden information. Additionally, if the document is retyped or if an Optical Character Recognition (OCR) software is used the hidden information can be destroyed (Roy and Manasmita, 2011; Singh, Singh and Saroha, 2009).



Source: (Roy and Manasmita, 2011)
Figure 3.6 – Line shifting

- **Word Shifting.** In this scheme, the embedded data is concealed by shifting the words horizontally, i.e. left or right to represent bit 0 or 1 respectively. Word shifts can be identified using a correlation method. This method however, does not detect much as the change of distance between adjacent words to justify a line of text is common. Nonetheless, if an observer is familiar with the algorithm of distances, he/she can compare the stego-object with the algorithm to discover the concealed message. Furthermore, the hidden information is destroyed if the document is retyped or scanned using an OCR application (Roy and Manasmita, 2011; Singh, Singh and Saroha, 2009).
- **Feature Coding.** In feature coding some of the features of the text are altered. For example, the height of some characters such as *'h', 'd'* or *'b*' are elongated or shortened a little or the extending or shortening of the horizontal line of the alphabet *'t'* can be used to conceal information. This method allows one to hide a large volume of information without making the reader aware of its existence. The shortcoming of this method is that the message is destroyed if the text cover is retyped or by the use of an OCR application (Agarwal, 2013; Roy and Manasmita, 2011).

3.6.3 Linguistic steganography

In this sub-section a comprehensive examination of linguistic (also referred to as Natural Language) steganography is presented. This is concerned with the lexical transformation of a cover medium in order to conceal a secret message rather than using the superficial properties of text (i.e. white spaces and word shifting). This assists in addressing the second research objective (OB2) which is to:

> **Identify existing linguistic steganalysis methods and determine its suitability for the detection of stegospam files generated using a probabilistic context-free grammar (PCFG) from non-stegospam files.**

This objective is accomplished through the analysis and review of the different linguistic encoding schemes, transformation techniques and its related stegosystem (i.e. a collective noun used to describe the stego-object and the steganalysis system). The transformation techniques implemented in linguistic steganography include Synonym Substitution, Syntactic

transformation and Semantic transformation which are discussed below (Ahvanooey, et al., 2019; Chang and Clark, 2014):

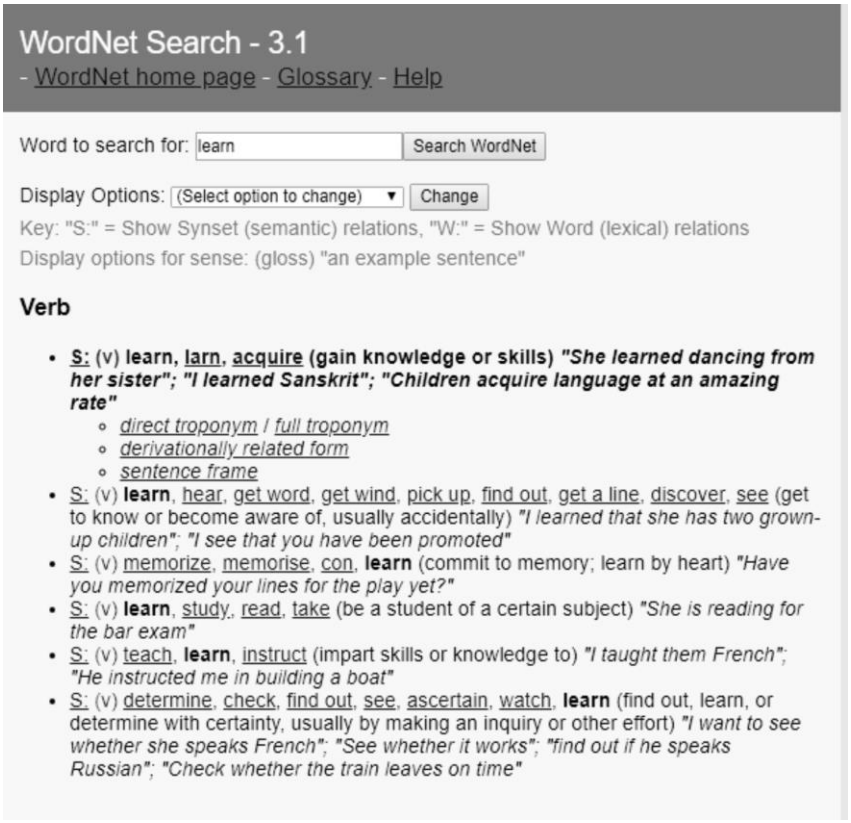
- **Lexical Transformation.** Lexical and phrase transformations refer to the process of substituting similar words or synonyms from a dictionary that has the same meaning within a specific context. Online dictionaries such as WordNet, VisuWords and Google Dictionary are designed to capture the lexical relationships between words and can be used to perform synonym or phrase substitutions. These dictionaries categorise English nouns, verbs, adjectives and adverbs into synonym sets known as synsets. Words in the same synset have the same or similar meaning and in principle can be substituted with one another (Chang and Clark, 2014; Xiang, et al., 2018).

  In lexical or synonym substitution, words are replaced by those with similar meaning (synonyms) to conceal sensitive information within a text-cover medium. These synonyms could represent a particular encoding scheme that only the sender and recipient knows about (Changder and Majumder, 2013).

  There are several challenges that are encountered when implementing synonym substitution. The first is word-category disambiguation, which constructs a word using a particular POS (part-of-speech) based on both its definition and context within the text corpus. For example, the word *'fast'* in the sentence *'Hold fast to the harness'* is an adverb, an adjective in the sentence *'he is a fast runner'* but a verb in the sentence '*Muslims fast during the month of Ramadaan'*. Therefore, by simply substituting a word with another that belongs to the same synset without considering its Part-of-Speech (POS) tag can expose a hidden message (Chang and Clark, 2014). The POS tagging is a technique supported by NLP and presents an opportunity for the first intersection between Natural Language Processing (NLP) and Steganography to realise the objective of this research study.

  The second problem, is word-sense disambiguation. This identifies the meaning of a word within a particular context. Words that belong to the same synset are said to be interchangeable and do not alter the meaning of the sentence hence they can be coded to conceal information. However, words which belong to more than one

synset cannot be used to conceal a secret message as they alter the meaning of a sentence and must be disregarded (Chang and Clark, 2014). Figure 3.7 below shows the synset for the word *'learn'* from WordNet 3.1.



Source: http://wordnetweb.princeton.edu/perl/webwn
Figure 3.7 – Synset of the word '*learn*' from WordNet 3.1

In this example, the word '*learn'*, '*larn'* and '*acquire'* belong to the same synset and are therefore interchangeable. If substituted they would not alter the meaning of a sentence. Therefore, the sentence '*Children acquire language at an amazing rate'* can be written as '*Children learn language at an amazing rate'* with not much semantic change. However, this cannot be written as '*Children instruct language at an amazing rate'* as firstly, the word '*instruct*' does not belong to the same synset as '*learn*' and secondly, it does not preserve the meaning of the sentence. This type of substitution, therefore, will arouse the suspicion of an observer. It should be remembered that the main purpose of linguistic steganography is to manipulate the cover text in such a manner that it is imperceptible and deemed innocuous by any observer (Chang and Clark, 2014).

The final problem is the suitability of a synonym in terms of the context of a sentence. For example, in the case of the synset for *chase, track, dog, tail, track, trail*, this synset would refer to aspects dealing with tracking and catching something. An incongruous sentence would result if the word '*chase*' were substituted with *'dog'* in a given sentence. Hence, it is important to verify the suitability of the synonym within the context of the text cover. To certify the appropriateness of a synonym, steganographers tend to implement an N-gram language model such as Google N-gram corpus (n-gram refers to a contiguous sequence of n items from a given sample of text or speech). This allows the system to select the most appropriate synonym based on the language model rather than randomly selecting one. These challenges are also applicable to text paraphrasing. Though, text paraphrasing may have more of an effect on the grammar of a sentence rather than the synonym (Chang and Clark, 2014). Consequently, synonym substitution remains one of the more popular and rudimentary techniques to linguistic steganography as discussed in Section 3.6.

- **Syntactic Transformation Methods.** Syntactic transformation methods are grounded on the fact that a sentence can be manipulated into several semantically equivalent syntactic structures through linguistic transformation techniques such as passivization[15], topicalization[16] and clefting[17] (Chang and Clark, 2014). Table 3.3 below demonstrates common syntactic transformations based on the English language including these three techniques of passivization, topicalization and clefting.

Transformation	Original sentence	Transformed sentence
Passivization	The dog kissed Peter.	Peter was kissed by the dog.
Topicalization	I like pasta.	Pasta, I like.
Clefting	He won a new bike.	It was a new bike that he won.
Extraposition	To achieve that is impossible.	It is impossible to achieve that.
Preposing	I like cheese bagels.	Cheese bagels are what I like.
There-construction	A cat is in the garden.	There is a cat in the garden.
Pronominalization	I put the cake in the fridge.	I put it there.
Fronting	"What!" Peter said.	"What!" said Peter.

Table 3.3 - Some common syntactic transformations in English
Source: (Chang and Clark, 2014)

[15] *Passivization refers to is the transformation of a sentence from an active form to a passive form (Nordquist, 2020)*
[16] *This is when the cause (a subject, word, or phrase) is made the topic of a sentence or discourse, typically by placing it first.*
[17] *A cleft sentence focuses on a specific part of a sentence to place emphasis to what we want to say.*

One of the earliest forms of syntactic transformation was presented by Atallah, McDonough and Raskin (2001) in that various sentences were created through the modification of the structural properties of a cover sentence. In preference to performing synonym substitution directly on the cover text, amendments were performed on the syntactic parse tree of the cover sentence to generate a sentence. When dealing with the syntactic structure of a sentence, therefore, the framework of the sentence needs to be understood. According to Noam Chomsky, in his book *Syntactic Structures* (1957), he explains that beneath every sentence constructed, there exists, in the mind of the speaker, an invisible, inaudible deeper structure. This deep structure is converted to a surface structure via transformational rules and the lexicon of a language (Atallah, McDonough and Raskin, 2001; Nordquist, 2020). A lexicon refers to the vocabulary of a language which is derived from the Greek word '*lexis*' meaning '*word*' or *'speech*'. Having a deeper understanding of sentence structures and syntax is critical when transforming a text cover medium into alternatives that are syntactically and semantically correct. Consequently, it improves the imperceptibility of the hidden message and the robustness of the cover medium.

Apart from syntactic transformations, Wayne (1995) has proposed a mimicry method related to linguistic syntax. It is used to generate stego-objects that integrate statistical properties similar to that of natural language. This method implemented a probabilistic grammar-based model to construct stego-objects. Since Wayne's and other mimicry methods placed emphasis on the syntactic structure of a sentence, they generally produced sentences that are nonsensical which were then easily perceptible to the human eye (Chang and Clark, 2014).

One example of a linguistic steganographic tool constructed on the mimicry method is NICETEXT, which examines the cover text to construct syntactic patterns and POS taggers to generate a set of '*sentence frames'*. Subsequently, through the use of a dictionary NICETEXT arbitrarily generates a series of words to construct a sentence. Although NICETEXT generates syntactically accurate sentences, they are almost always grammatically and semantically anomalous. Consequently, several

steganalysis algorithms have been developed to detect or classify text-based stego-objects generated using NICETEXT (Changder and Majumder, 2013; Koluguri, Gouse and Reddy, 2014).

- **Semantic Transformation Methods.** Semantic transformation involves the modification of a sentence to generate variants that communicate the same meaning as the original. This technique has been adopted in various fields including Natural Language Processing, Linguistic Sciences and Steganography. It is one of the more sophisticated techniques used for the implementation of linguistic steganography and requires the latest tools and knowledge on how to model natural language semantic representations in a manner that preserves the meaning of each sentence. In their paper Atallah, McDonough and Raskin, (2002), implemented semantic transformations which were used to generate alternate cover sentences through the modification of the original text without altering its meaning. This approach involved several techniques such as word substitution, trimming and grafting by sourcing information from ontological[18] semantic resources (Chang and Clark, 2014).

  Another semantic transformation technique used for the implementation of linguistic steganography is the machine-generated sentence. In this approach a translation system or algorithm is used to generate alternative sentences from a cover sentence. The advantage of this approach is that it is not perfect, hence it is difficult to determine whether an anomaly has originated from the translation system, the cover sentence, or the hidden text.

  A final approach to linguistic steganography is sentence compression which was introduced by Dorr, Zajic and Schwartz in 2003. This approach involves the reduction of the sentence by removing unnecessary adjectives in noun phrases for example in the sentence: '*he spent only his own money*' can be reduced to '*he spent his own money'*. This reduces the sentence but does not alter the semantics of the sentence. In order to verify that the deletion is grammatically and semantically correct it can be certified against an N-gram linguistic corpus such as Google N-gram and against a

[18] *"A linguistic ontology is a formal knowledge representation of the world; a conceptualization of entities, events and the relationships in an abstract way"* - (Chang & Clark, 2014)

Support Vector Machine (SVM) model that combines N-gram statistics and lexical association (Chang and Clark, 2014).

For this research study, therefore, it is important to understand the linguistic ontology of a natural language to develop and propose a steganalysis system that is capable of detecting stego-objects. It is one of the primary objectives of steganographers to generate cover sentences that are semantically accurate, possess the statistical properties that accurately mimic natural sentences, and are syntactically correct. Additionally, these factors can exponentially improve the imperceptibility and robustness of a hidden message within a cover medium.

This section discussed the different forms of text-based steganography and linguistic transformation techniques used to embed a secret message within a plaintext cover medium. Furthermore, it assisted in addressing the secondary research question SQ2:

Which linguistic steganographic techniques are favoured in order to embed a secret message within plaintext files?

The next section discusses the different encoding algorithms employed by these linguistic steganographic approaches to generate a seemingly innocuous cover text.

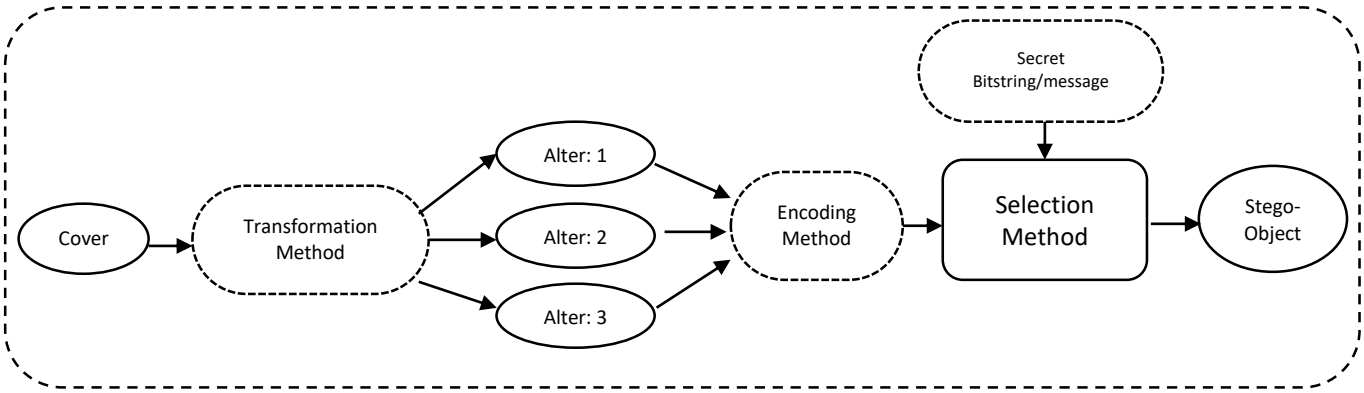
3.7 Linguistic Steganographic Encoding Algorithms

The previous section presented a discussion on the numerous linguistic transformation techniques for generating alternative text-based cover mediums from a given input text. Once this process (referred to as the Linguistic Transformation Model) is completed it moves onto the encoding process. Figure 3.8 below illustrates an overall perspective on the Linguistic Stegosystem or process. The encoding model is used to map each derivative of the input text into a code that can be used to represent a secret bitstring or message (Chang and Clark, 2014).

In this section, three linguistic encoding methods are discussed that are commonly used to encode a secret message within a plaintext cover medium. This section is imperative as it assists in understanding the different encoding techniques implemented during the linguistic

steganographic process, consequently supporting the development of a solution that may assist in detecting them.

To understand these encoding algorithms assume the cover text reads: '*This is a wonderful place'* and synonym substitution as the elected transformation method that will substitute the word '*wonderful*' with one of the synonyms in the synset X = (*nice, great, wonderful, decent*). Similarly, the encoding algorithms discussed in this section have been implemented in other linguistic transformation methods such as those discussed in the previous section.



Source: (Chang & Clark, 2014)
Figure 3.8 – Three modules in a linguistic stegosystem

Binary Equivalents

Once again, we can assume the synset *X* = {nice, great, wonderful, decent} and the associated binary equivalents as shown in the Table 3.4 below.

Word	***Binary Equivalent***
Nice	00
Great	01
Wonderful	10
Decent	11

Table 3.4 – Lexical substitution using Binary Equivalents

In this encoding scheme the binary equivalent is calculated and used by co-conspirators to encode and decode a secret message within a plaintext cover medium. This sort of concealment can be easily injected into an innocuous spam email. Once the secret message is deeply embedded within the text cover medium it becomes imperceptible to an observer.

Therefore, the sentence '*This is a wonderful place'* will produce a binary code $(10)_2$ and '*This is a decent place'* will produce the binary code $(11)_2$ – which are then interpreted by the co-conspirators (Chang and Clark, 2014; Koluguri, Gouse and Reddy, 2014). In this scheme the cover text is seen to be subjected to lexical or synonym substitution prior to the execution of the encoding scheme.

Block Code Algorithm

For a given set with a cardinality *n*, the block code method assigns *m*-bit binary codes from 0 to $(2m - 1)$ to the elements within the set, where $2m \leq n$. Therefore, the synset X has a cardinality = 4, the block code method can assign either one or two bit codes as shown in Table 3.5 (Chang and Clark, 2014).

1-bit	2-bit	Synonym
0	00	Nice
1	01	Great
0	10	Wonderful
1	11	Decent

Source: (Chang and Clark, 2014)
Table 3.5 – Example of the Block Code method

Therefore, a steganographer could choose to encode any or all of the generated text-based cover mediums by using the associated synonyms for the word *wonderful* to embed a secret message. It is important to notice that in this method, a steganographer may opt to implement a 1-bit or 2-bit encoding system as shown in Table 3.5. One of the major drawbacks of the 1-bit system arises in its payload capacity. This is seen in the case of *nice* and *wonderful* where both words represent the code 0 and *Great* and *Decent* representing the code 1, hence, limiting the amount of information that can be coded. However, in the case of a 2-bit encoding system we are able to produce four unique codes, one for each element of the synset. Therefore, in this case the payload capacity is higher (Chang and Clark, 2014; Koluguri, Gouse and Reddy, 2014).

Mixed-Radix Number Algorithm

This algorithm is very similar to the block code algorithm with the exception that the binary equivalents are concealed by mixed radix numbers (Chang and Clark, 2014). To elaborate, let us assume that we want to increase the payload capacity of the cover sentence. This is achieved by introducing synset $y = \{place, area\}$. Since disclosure the binary equivalents is not wanted, the synonyms using radix numbers are encoded. Since synset y has a cardinality of 2, we assume a base of 2, therefore the radix numbers will be $y = \{place = 0_2, area = 1_2\}$. For synset x, which has a cardinality of 4 we assume the base of 4. Therefore, the radix numbers will be $x = \{nice = 0_4, great = 1_4, wonderful = 2_4, decent = 3_4\}$. The mixed radix and binary equivalents are illustrated in the Figure 3.9.

Code	Word	Binary
$\mathbf{0_4}$	Nice	00
$\mathbf{1_4}$	Great	01
$\mathbf{2_4}$	Wonderful	10
$\mathbf{3_4}$	Decent	11

Code	Word	Binary
$\mathbf{0_2}$	place	O
$\mathbf{1_2}$	area	1

Source: (Chang & Clark, 2014)
Figure 3.9 – Mixed radix with binary equivalents for Synset X and Synset Y

Now that the tables are established, it can be assumed that the secret bitstring is 110_2 which has the decimal value of 6. Using the mixed radix system, it can be encoded as ($3_4 0_2$), which is represented by the cover sentence – *'this is a decent place'*. This is because the radix position of 3_4 represents the word decent which in turn represents the binary string *'11'* and the radix position 0_2 represents the word place which in turn represents the binary equivalent of *'0'*, joining them gives us 110_2, representing the decimal value of 6. For a second example, it can be assumed that the secret bitstring is 111_2, which has a decimal value of 7. Using the mixed radix system, it can be encoded as ($3_4 1_2$), which is represented by the cover sentence *'this is a decent area'* (Chang and Clark, 2014; Koluguri, Gouse and Reddy, 2014).

Huffman Code Algorithm. In this approach a steganographer initially encodes the words of a synset using the Huffman Code algorithm to calculate the variable-length code based on the relative frequency of each word within the synset. This frequency refers to the words that

have a higher probability of being chosen by the stegosystem. Once again, the objective is concealment, therefore, to improve the imperceptibility and robustness of the stego-object the Huffman Code algorithm assigns a shorter code for words that have a higher frequency as opposed to words that have a lower frequency. Words with a lower frequency and a longer associated code are referred to as leaf nodes. This ensures that the derivatives produced by the algorithm are of a better quality and have a higher statistical property that is closer to a natural language such as English (Bajpai and Saxena, 2013; Chang and Clark, 2014). The encoding process for the cover sentence '*we finish the charitable project'* is illustrated in Figure 3.10.

We

Code	Word	Probability
0	complete	0.77
1	Finish	0.23

the charitable

Code	Word	Probability
110	Labour	0.05
0	Project	0.69
10	Task	0.25
111	Undertaking	0.01

Source: (Chang and Clark, 2014).
Figure 3.10 – An example of the Huffman Code method.

In Figure 3.10 above, two tables represent two synsets. In the first table the word '*complete'* has a higher probability due to its frequency and is therefore assigned the shortest-length binary code of '*0'*. Likewise, in the second table the word '*project'* has a higher probability amongst the words in the synset and is therefore assigned the shortest-length binary code of *'0'*. Subsequently, using the Huffman Coding algorithm the cover sentence that will most likely be generated is '*We complete the charitable project*' (Bajpai and Saxena, 2013; Chang and Clark, 2014).

A fundamental threat to any stego-object is detection. Therefore, a strong encoding algorithm is key to ensuring the imperceptibility, survivability and robustness of any stego-object. Consequently, the encoding algorithms discussed in this section are commonly integrated into several linguistic steganographic tools such as Spammimic, NICETEXT and Texto through the use of various natural language transformation methods mentioned in the preceding section. The next section discusses Linguistic Steganalysis.

3.8 Linguistic Steganalysis

As previously defined, steganalysis refers to the process that involves the discovery of the existence of hidden information within a cover medium, in other words, detecting stego-objects (Richer, 2019; Shih, 2017). Given the rationale of this research study, this section focuses specifically on the steganalysis methods associated with Linguistic or Natural Language (NL) steganography. Gaining an understanding of the different linguistic steganalysis techniques is essential in identifying the work done and challenges in the field, and guides the proposal of a linguistic steganalysis that aims to test the hypothesis of this research study. Additionally, this section addresses the secondary research question SQ3:

> **Which steganalysis techniques are commonly employed for the detection of linguistic steganography?".**

Whilst there have been several steganalysis methods proposed and implemented over the past few decades, many have been directed towards audio, image and video steganography and very little towards linguistic steganography (Din, Samsudin and Lertkrai, 2012; Richer, 2019). The application of a steganalysis can be classified into two categories; namely, linguistic steganalysis (which focuses on the discovery of steganographic traces hidden in complex linguistic structures which is the focus of this study), and digital steganalysis (which focuses on the discovery of steganographic traces in image, audio, and video cover mediums). Existing linguistic steganalysis methods, however, are comparatively weaker when compared to its well-established digital counterparts (Din, Samsudin and Lertkrai, 2012).

Generally, a steganalysis process comprises two techniques; visual analysis and statistical analysis (Shih, 2017). Visual analysis involves comparing a suspected file against the original which aims to expose the presence of a concealed message through inspection, either by the human eye or with the aid of a computer system. Whilst this approach is simple, it is not very effective since, most of the time, the original file is unavailable. Statistical analysis, on the other hand, detects statistical vicissitudes in the cover medium through the use of statistical algorithms to determine if its statistical properties deviate from the norm. Hence, it aims to detect even the slightest variation to the statistical balance of a file that may have been inflicted by a steganographic embedding or transformation process (Shih, 2017).

The robustness of a stego-object is its ability to withstand a variety of attacks and still maintain the integrity and the imperceptibility of the hidden message and linguistic steganography is no different (Shih, 2017). Generally, steganalysis attacks involve either detection, modification or destruction of the embedded message. In linguistic steganalysis, detection may be achieved through a statistical attack where the steganalysis process equates the word frequency distribution of a stego-object to that of a similarly trusted file to detect anomalies or through visual inspection in which visual anomalies are identified by the human eye or via computer-aided software. A modification attack refers to the modification of the file format of a stego-object or by compressing the stego-object to reduce any redundancies subsequently distorting the embedded message. Lastly, the destruction of a stego-object would mean deleting the suspected stego-object (Shih, 2017: Xiang, et al., 2018).

3.8.1 Steganalysis using statistical analysis

In linguistic or natural language steganography, synonym substitution has emerged as the preferred transformation method when generating alternative cover texts. As previously shown in Figure 3.8 (three modules in linguistic stegosystem) during the information transformation and embedding processes, the statistical and linguistic characteristics of a text cover medium are inevitably altered, which serve as key indicators for the steganalysis (Xiang, et al., 2018). This is evident in the synonym substitution steganographic system named 'Tyrannosaurus Lex' (TLex) developed by Keith Winstein in 1998, which implemented a multi-base synonym substitution technique using synonyms extracted from WordNet[19] to generate alternative cover texts (Taskiran, et al., 2006). This approach proved ineffective as the stego-text it produced contained several semantic, pragmatic and statistical problems. The fundamental objective of any steganalysis is to extract detection features that can assist in comprehensively detecting any modifications caused by the embedding or transformation processes (Xiang, et al., 2018).

Consequently, steganographers and researchers started developing improved synonym substitution techniques to generate fluent stego-texts that exhibited grammatically and semantically correct synonym transformations such as the Vertex Coding Method proposed

[19] *https://wordnet.princeton.edu/*

by Chang and Clark in their research paper – 'Practical linguistic steganography using contextual synonym substitution and a novel Vertex Coding Method' (Xiang, et al., 2018). Consequently, linguistic steganalysis methods that employ statistical algorithms search for statistical anomalies in a cover text which serve as an indicator for the presence of a stego-object. These three statistical approaches are discussed below (Din, Samsudin and Lertkrai, 2012) as Inverse Document Frequency (IDF); Relative Frequency Probability (RFP); and the Neural Network Language Model (NNLM).

- **Inverse Document Frequency (IDF).** One of the more common statistical techniques adopted in linguistical analysis is the N-gram analysis approach which is also implemented by several steganalysis for the classification of stego-objects generated using synonym substitution as discussed in this section and in greater detail in Chapter 5, Section 5.6 – Related Research (Xiang, et al., 2018). An N-gram is defined as a continuous sequence of *n* items from a sample of text. Simply put, an N-gram is a sequence of words: a 2-gram *(or bigram)* is a two-word sequence of words like '*thank you'* and a 3-gram *(trigram)* is a three-word sequence of words like '*start to read'* (Jurafsky and Martin, 2018). Therefore, this method extracts features by estimating the N-gram probabilities between the original unmodified and the steganographically modified sentences. Once the N-gram probabilistic estimates are determined, its context information is calculated using the Inverse Document Frequency (IDF) algorithm, which is used to measure the appropriateness of a synonym within its context by producing statistical values that are used to distinguish between normal and stego text (Enge, 2015; Xiang, et al., 2018). The IDF is a measure of determining the frequency of a word within a dataset of text. The IDF of a particular word is determined by using a logarithmic calculation. It is the ratio of all existing texts and documents of an entire dataset and the number of texts that contain the defined keyword (Enge, 2015). , This is represented by the following equation:

$$IDF_t = \log\left(1 + \frac{N_D}{f_t}\right)$$

Source: (Enge, 2015)
Figure 3.11 – Inverse Document Frequency Formula

With:

- N_D = Total number of documents in the given text corpus
- f_t = frequency of a word or term. Number of documents where the term ***"t"*** appears.
- Note: ($f_t \neq 0$) which implies that the term is not in corpus.

The IDF value increases proportionally to the number of times a word appears in a particular document, but is counterweighed by the frequency of the word in the corpus, which implies that some words are used universally across various documents within the corpus. The IDF is commonly used in a number of NLP techniques such as text mining, search queries and a summary, which are further addressed in Chapter 4 (Natural Language Processing) (Crosson, 2016).

- **Relative Frequency Probability (RFP).** Another method of establishing statistical frequencies of a synonym found within a document corpus is known as Relative Frequency Probability (RFP). This method sorts synonymous words in descending order based on their frequencies within a large dataset to establish a synonym vector. Each synonym has its own attribute pair which contains its position in a synonym vector and the vector's dimension. An example of this is shown in Table 3.6 below (Relative Frequency Probability Matrix). The relative frequencies of the attribute pairs appearing in the detected text are computed for the extraction of detection features. The RFP demonstrated an improvement in the detection accuracy when compared to IDF (Xiang, et al., 2018). Whilst these statistical techniques (IDF and RFP) can differentiate stego texts from natural texts through the analysis of rudimentary features, they are incapable of detecting deep word semantics and relations. Therefore, to improve the performance and accuracy of a linguistic steganalysis method, it must incorporate deep semantic characteristics and relationship of words by using a much larger learning corpus. One such approach is the implementation of a Neural Network Language Model (NNLM) (Xiang, et al., 2018), which is discussed in the next sub-section.

Table 3.6 below illustrates an example of the Relative Frequency Probability Matrix.

Word	Frequency	Relative Frequency
Car	7	7/20 = 35%
Bus	3	2/20 = 15%
Train	5	5/20 = 20%
Aeroplane	5	5/20 = 20%
Total	20	20/20 = 100%

Table 3.6 – Relative Frequency Probability Matrix

- **Neural Network Language Model (NNLM).** Before examining Neural Network Language models, cognisance needs to be made of a few fundamental terms related to language models. A language model is a learning algorithm which captures the salient statistical characteristics for a set of words of a natural language, typically allowing for probabilistic estimates for the next word given the preceding ones (Bengio, 2008). Furthermore, the NNLM is a language model which harnesses the capability of neural networks to learn the semantic relationship between a synonym and the words within its context (Xiang, et al., 2018). Additionally, this assists the language model to significantly improve its 'dimensionality'. Dimensionality refers to the need for a large training corpora to learn extremely complex functions such as those in natural language processing. This is made possible due to the enormity of resources and dictionaries available on the internet which are essential for improved performance and accuracy. A distributed representation of a word is a vector of features which characterise the meaning of the word. Therefore, in choosing the features of a word, one might choose the grammatical features like gender or plurality, as well as semantic features like animate or invisible. With a neural network language model, one relies on its learning algorithm to discover these features (Bengio, 2008; Turian, Ratinov and Bengio, 2010).

  The implementation of a NNLM in statistical analysis can significantly help in unveiling the deep and implicit semantic relationships between a synonym and its context words. Through this process content-fitness values for synonymous words are

produced, which are then fed into a classifier such as a Support Vector Machine (SVM) or Naïve Bayes to help distinguish stego-text from cover-text (Xiang, et al., 2018).

An SVM is fundamentally a set of related supervised machine learning algorithms that analyses data and identifies patterns used for classification and regression purposes. The SVM is based on the concept of establishing a hyperplane that divides a dataset into classes. The SVM is commonly used in Machine Learning and Natural Language Processing tasks and is discussed in greater detail in the next chapter (Bambrick and Aylien, 2016; Nechta and Fionov, 2011).

To help generate the 'content-fitness' values and improve the accuracy of linguistic steganalysis, several Neural Network Language models incorporate NLP techniques such as word representation, word embedding and brown clustering. These techniques help reduce the number of false positives when classifying a text corpus (Xiang, et al., 2018).

Each of these techniques are briefly discussed here but in greater detail in the following chapter:

- ***Word Representation.*** This is a mathematical object that is associated with each word or vector with corresponding semantic and grammatical interpretations. In terms of linguistic steganalysis, word representation allows us to capture the semantics of natural language words, especially synonymous words. This technique has emerged as a popular feature in machine learning and natural language processing and has significantly improved the performance of text classifiers such as SVM (Xiang, et al., 2018; Turian, Ratinov and Bengio, 2010).
- ***Word Embeddings.*** This technique exposes context information and contains rich semantic relations, hence the suitability of a synonym within its context can be efficaciously measured through the application and analysis of word embeddings. Word embeddings are a dense, low-dimensional and real-valued representation typically constructed using a neural language algorithm such as Word2Vec and Doc2Vec (discussed in greater detail in Chapter 4) (Turian,

Ratinov and Bengio, 2010). This technique is largely applied in NLP and can significantly benefit linguistic steganalysis as demonstrated by Xiang et al. in their paper (Xiang, et al., 2018) in which it was proposed that a steganalysis technique implement word embeddings as one of the statistical methods to ascertain the semantic distances between two words and the word correlation between a synonym and each word within its context. (This approach is discussed in greater detail in Chapter 5, Section 5.6 – Related Research).

- ***Brown Clustering.*** This represents a hierarchical clustering algorithm which group words based on the context in which they occur. This allows language algorithms to take full advantage of the mutual information of bigrams therefore, it is known as a Cluster N-gram model (Turian, Ratinov and Bengio, 2010). One of the key advantages of this approach is that it addresses the data sparsity problem inherent in many language models by compensating weaker clusters with stronger clusters at several levels in the hierarchy (Turian, Ratinov and Bengio, 2010).

3.8.2 Steganalysis using visual analysis

In any steganalysis, statistical analysis is usually the preferred method as a perceptual approach to the detection of negligible variations in text-based cover mediums can be an intimidating task as in the case with visual analysis. Visual analysis requires careful observation for any modification of synonyms, semantic paraphrasing, lexical changes, and rhetorical changes. Whilst this approach is challenging and time-consuming there are instances where steganalysts do adopt visual inspection on suspected files in combination with statistical analysis methods (Ahvanooey, et al., 2019).

One of the objectives of visual analysis is to discover patterns in a corpus that may lead to the identification or signature of a particular steganographic tool or even the detection of concealed information. Visual analysis is generally conducted by comparing patterns of suspected stego-objects against the original file. This approach is termed the 'known-cover' attack (Ahvanooey, et al., 2019; Richer, 2019).

In the event the original file is not available and the suspected stegotext was generated using a PCFG (such as NICETEXT, TEXTO or Spammimic) visual analysis can be implemented to compare a sizeable sample of the stegotext to discover possible signatures that may identify the steganographic tool employed or perhaps infer the presence of a concealed message (Richer, 2019). In terms of this research study, visual analysis will not be considered as part of the proposed steganalysis; however, the brief discussion presented was essential in providing a holistic view of the different steganalysis techniques.

3.9 Chapter Conclusion

The application of steganography was initially intended for security purposes such as watermarking and the concealment of classified or personal information. However, over the past few decades, digital steganography has been exploited by cyber-criminals to covertly communicate with co-conspirators. An instance of this is presented in section 2.8 of Chapter 2, where a husband covertly conspired with a criminal to murder his wife by using Spammimic to randomly generate spam covers with the concealed message, hereinafter referred to as stegospam (Yu, 2015).

One of the most difficult steganographic approaches is linguistic steganography, which is a branch of text steganography and a focus of this research study. Linguistic steganography has proven to be a feasible option for steganographers and hackers alike as it has a high degree of imperceptibility when compared to other types. Whilst its payload capacity is smaller than other cover mediums, this directly contributes to its imperceptibility and survivability. These are critical factors that provide the motivation for the implementation of linguistic steganography when electing spam email as the cover medium.

Section 3.4 and 3.5 addresses the secondary research sub-question SQ2:

Which linguistic steganographic techniques are favoured in order to embed a secret message within plaintext files?

and Section 3.6 addresses the secondary research sub-question SQ3:

Which steganalysis techniques are commonly employed for the detection of linguistic steganography?

Additionally, this chapter provides some insight into the research objective OB2:

Identify existing linguistic steganalysis methods and determine its suitability for the detection of stegospam files generated using a probabilistic context-free grammar (PCFG) from non-stegospam files.

A further examination on this objective is presented in Chapter 5, Section 5.4.5.

The next chapter provides a discussion on Natural Language Processing.

4

Natural Language Processing (NLP)

This chapter presents a detailed discussion on the various Natural Language Processing concepts and methods to address the research objective – 'Identify and describe various Natural Language Processing methods or algorithms that may be considered for the proposed steganalysis'.

Introduction

In this chapter the concept of Natural Language Processing is discussed. This is a discipline of Artificial Intelligence that combines the studies of computer science and computational linguistics (Khurana, et al., 2017). It is a complex and challenging field that strives to bridge the gap between the manner in which a computer and a person interprets information. Natural language processing is defined by Liddy (2001), as:

> [A] theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications.

Consequently, NLP has been successfully implemented in the fields of computer security, medical science, accounting, and linguistics. In this chapter the different approaches to natural language processing are investigated to determine its ability to detect or classify stego-objects generated using linguistic steganography. Furthermore, it will contribute to addressing the research objective stated above.

Chapter 3 presented a few natural language processing techniques such as Word Stemming, Word Representation, and Part-of-Speech tagging and provided a discussion how these techniques were integrated into steganographic tools to generate text-based stego-objects. In this chapter, NLP, its methods and applications are discussed. Natural language processing

is implemented in various tasks such as Document Generation, Information Extraction, Auto-Summarization, and Machine Translation (Khurana, et al., 2017). The focus for this research study is on Information Extraction as the purpose of IE is to excerpt the context and pragmatics[20] of a text which is required for the feature extraction and classification of stego-objects for the proposed steganalysis.

The discipline of NLP is classified into two major categories; namely, Natural Language Understanding (NLU) and Natural Language Generation (NLG) as illustrated in Figure 4.1 – *A* Broad classification of NLP and discussed in Section 4.1 (Khurana, et al., 2017). Based on the research objective stated earlier, this chapter will focus on NLU.

As stated, Natural Language Processing combines the disciplines of Computer Science and Linguistics. Hence, Section 4.2 discusses the different phases of linguistic analysis in common with the appropriate NLP methods discussed in Section 4.3. In doing so, we are able to take cognisance of what each method endeavours to achieve linguistically at each phase. Consequently, affording the opportunity to develop a suitable steganalysis approach that is capable of analysing plaintext spam files to detect traces of linguistic steganography, hence testing the research hypothesis.

Section 4.3 presents a methodical approach to natural language processing through the discussion of an NLP pipeline. It is crucial to analyse and understand the NLP methods associated with each phase of the pipeline; such as sentence and word segmentation, lemmatization, and Dependency Parsing as they are important considerations for the proposed steganalysis. Section 4.4 introduces and discusses key feature extraction algorithms such as Word2Vec and Doc2Vec which are implemented to mine deep and meaningful features from a text corpus. Section 4.5 presents a detailed discussion on the classification of text through the use of different Machine Learning (ML) or classification models.

Section 4.6 presents a discussion on the application of NLP in cyber-security for the detection of various forms of attacks such as phishing, domain generation algorithm classification and spam filtering. Lastly, this chapter investigates the implementation of NLP techniques in

[20] *Pragmatics refers to the fundamental or basic form of a sentence or word and discussed in Section 4.2.*

existing linguistic steganalysis research to determine their suitability for the proposed solution as required by the research objective.

The structure of this chapter is as follows:

Section 4.1 – Taxonomy of Natural Language Processing (NLP).

Section 4.2 – Natural Language Understanding (NLU).

Section 4.3 – Implementation of NLP in Linguistic Analysis.

Section 4.4 – Feature Extraction.

Section 4.5 – Text Classification using Classification Models.

Section 4.6 – Application of NLP in Cyber-Security.

Section 4.7 – Related application of NLP in Linguistic Steganalysis.

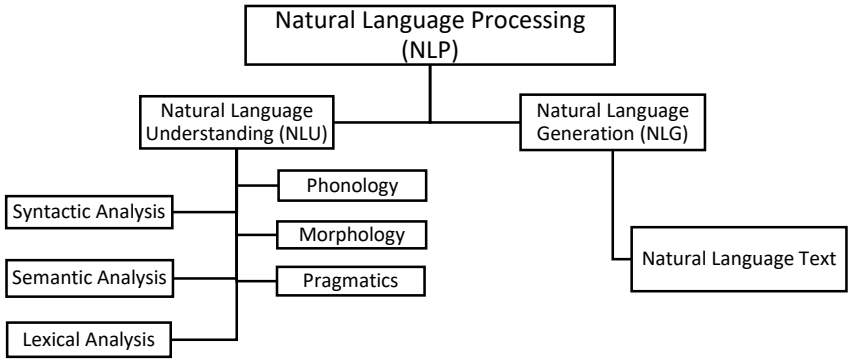
4.1 Taxonomy of Natural Language Processing

As stated above, NLP is a branch of Artificial Intelligence that incorporates linguistic analysis and computational methods which are committed to making computers 'understand' statements or words written in a natural language such as English (Singh, 2018). Natural Language Processing can be classified into two major categories i.e. Natural Language Understanding and Natural Language Generation (Khurana, et al., 2017). Natural Language Understanding is the primary objective of NLP and is yet to be fully realised. For a computer system to truly 'understand' a natural language it must be able to:

- rephrase an input text;
- translate the text into another language;
- answer questions based on the contents of the text; and
- draw inferences from the text.

Even though there have been major developments with regard to these challenges, there is still a lot of work that needs to be done. Natural Language Generation, on the other hand, refers to the automatic generation of text that complies with the syntactic, pragmatic and semantic rules of a natural language and is commonly used in summarisation tasks and the generation of context-free documents (Liddy, 2001; Singh, 2018).

To understand the process of NLP and how it can be implemented in this research study we need to first understand the discipline of linguistics. Central to the study of linguistics is the language itself which is defined as a set of rules, sounds or symbols that, when combined in a conventional manner, can convey a message. Consequently, linguistics is the study of language which comprises of its Phonology, Morphology, Lexical, Syntax, Semantics and Pragmatics (Khurana, et al., 2017), as illustrated in Figure 4.1 below of a Broad Classification of NLP.



Source: (Khurana, et al., 2017)
Figure 4.1 – A broad classification of NLP

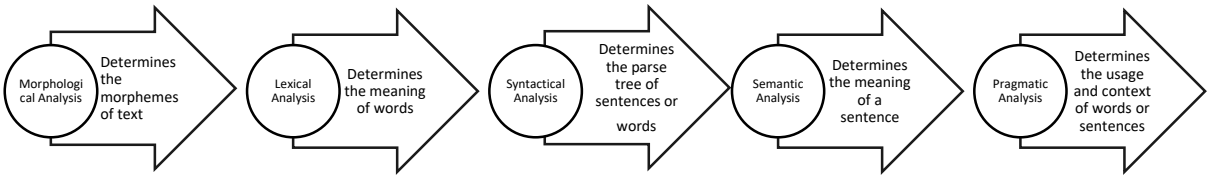
As mentioned, NLP is divided into two categories: Natural Language Generation and Natural Language Understanding. Natural Language Generation was first introduced in Chapter 3, section 3.4.1 (Random and Statistical Generation) which presented a discussion on the random generation of context-free text-based cover mediums that adhere to the statistical, syntactical and semantic rules of a natural language. However, NLG has several limitations which is evident in the pragmatics of the cover text and the poor correlation between the derived sentences. Consequently, many steganalysts have developed several steganalysis techniques that exploits this limitation (Koluguri, et al., 2014; Zhi-li, et al., 2008). This research study, however, is not concerned with the auto-generation of text covers, rather it is concerned with the decomposition and analysis of existing bodies of text, which is the focus of Natural Language Understanding and discussed in the next section.

In addition, a rigorous understanding of linguistic concepts and NLP techniques is crucial for the design and development of an NLP-based steganalysis that can efficaciously detect linguistical anomalies in a given text that may indicate the presence of an embedded message. Furthermore, it is equally important to review existing research or work done in this field as it will provide further insight into the research objective stated earlier.

4.2 Natural Language Understanding

Most of the effort in NLP is piloted by computer scientists, who frequently intersect with other disciplines including Linguistical Computation. Therefore, to develop algorithms that can analyse and process a natural language, software developers and computer scientists must understand the different levels and concepts of a natural language and how they can be coordinated to conduct linguistical analysis on a text corpus or to generate meaningful and syntactically accurate text (Chowdhury, 2003; Khurana, et al., 2017). Existing NLP tasks such as automatic text summarisation, co-reference analysis, and context-free text generation implement algorithms that handle linguistical tasks such as semantic, syntactic, pragmatic, lexical, and morphological analysis which are discussed in the following sections (Garbade, 2018; Khurana, et al., 2017).

As mentioned, a sound understanding of these concepts is crucial in the design and development of an NLP-based steganalysis that will allow this study to address the research objective and test the research hypothesis. Figure 4.2 below outlines the different levels involved in linguistic analysis. Consequently, these levels are implemented in the proposed steganalysis discussed in Chapter 5 (Research Design and Methodology).



Source: (Alyafei, 2018)
Figure 4.2 – Phases of Linguistical Analysis

4.2.1 Morphological analysis

Morphology, a sub-discipline of linguistic sciences refers to the study of the internal structure of words. This level focuses on the computational nature of words, which is composed of morphemes - the smallest unit of meaning (Khurana, et al., 2017). For example, the word '*re-registration'* can be morphologically analysed into three morphemes: the prefix '*re'',* the root '*register'* and the suffix '*tion'*. By deconstructing a word into its constituent morphemes, one is able to understand its root meaning or lexical sense. Similarly, NLP programming libraries such as CoreNLP from the Stanford group, SpaCy, and NLTK are able to derive the core word and its meaning through the implementation of numerous decomposition algorithms which are discussed in the following sections (Liddy, 2001; Vladimir, 2018).

Considering that the analysis and comprehension of a human language is the foremost objective of NLP, morphological analysis should ideally be the first step which entails the process of breaking down a word or sentence into its most granular form or morpheme. Natural Language processes such as Lemmatization, Stemming and Part-of-Speech tagging are typically used to determine the morphology of a word and serve as important processes for all NLP solutions including this research study (Prabhakaren, 2018; Shrivastava, et al., 2004). Hence, they are discussed in the following sections.

4.2.2 Lexical Analysis

Lexical analysis is a process that focuses on interpreting the meaning of individual words relative to their etymological meaning (i.e. the origin and historical development of a word and its meaning) together with its corresponding Part-of-Speech[21] (POS) tag. This process follows after morphological analysis. Lexical analysis implements the language's lexicon – which is a collection of lexemes that represents its fundamental unit of lexical meaning. Lexical analysis is an essential process in linguistical analysis and a significant step for several NLP solutions (Espanol, 2017; Liddy, 2001).

Seeing that the overall objective of lexical analysis is to interpret the meaning of words, one must take cognizance of the fact that certain words at this level may be interpreted differently based upon its context and can, therefore, be represented by more than one POS tag.

[21] *Part-of-Speech tag refers to whether a word is a noun, verb, adjective, proper noun etc.* (Geitgey, 2018).

Therefore, such a word could be identified as a noun or a verb subject to its context. An example of which could be the word *'race'*. In the sentence *'a race amongst all competitors'*, the word *'race'* represents a noun which refers to a competition, and in the sentence *'he raced home!'* it represents a verb, implying that he hurried home. From a human perspective it is a lot easier to identify the meaning of the word as we may understand its ontology and context, but from a computational linguistics or NLP perspective, it is a lot more challenging (Espanol, 2017; Khurana, et al., 2017; Liddy, 2001).

Traditionally lexical analysis was handled manually by experienced linguists timeously sorting and associating the meaning of words, its relations and synonyms. However, with the proliferation of the internet and online dictionaries such as WordNet[22] and Dante[23], computational linguistics and natural language processing has become a lot easier (Bird, Klein and Loper, 2009). WordNet in particular has been implemented in several NLP solutions to perform various tasks. WordNet groups English words into sets of cognitive synonyms known as synsets, each expressing a distinct concept. Synsets are interconnected by means of conceptual-semantic and lexical relations (Fellbaum, 2006)

Lexical analysis is an essential step in any linguistical analysis and natural language processing task. Once the lexical meaning of a word is determined it is then subjected to semantic, pragmatic, and syntactic analysis to support a system in *'understanding'* a word or sentence of a natural language. These processes are essential in this research study as they and their corresponding NLP algorithms provide insight into the design and development of the proposed steganalysis.

4.2.3 Syntactic analysis

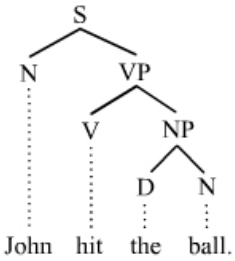
This level of a language examines the words within a sentence in order to establish its grammatical structure also known as its syntax. In linguistic sciences, syntax focuses on the order and dependency of words which constitutes its meaning (Khurana, et al., 2017). Subsequently, the morphology of a word, syntax, and its context are essential considerations when attempting to discover evidence of linguistic steganography within a cover medium.

[22] *https://wordnet.princeton.edu/*
[23] *https://euralex.org/publications/the-dante-database-database-of-analysed-texts-of-english/*

Syntactic anomalies generated by word or phrase substitution are key indicators for the probability of concealed messaging as discussed in Chapter 3 (Chang and Clark, 2014).

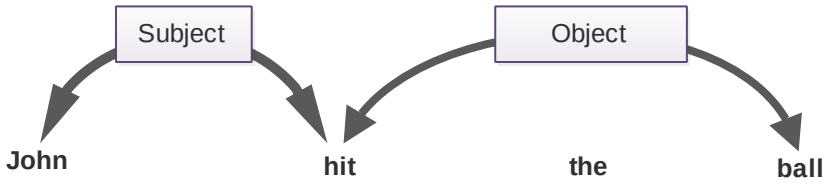
In terms of the syntax of a sentence, NLP employs a technique known as 'parsing' (a process of analysing a string of symbols) which is classified into two categories based upon their grammatical formalism: Constituency Parsers and Dependency Parsers (Singh, 2018). Constituency parsers, also known as a 'Phrase Structure Tree', produce syntactic analysis in the form of a tree that demonstrates the phrases comprising of a sentence and the hierarchy in which the phrases are associated as demonstrated in Figure 4.1 below for the sentence '*John hit the ball'*. This process is commonly modelled by Context-Free Grammars (CFG) for pronoun resolution, labelling phrases, and semantic roles. Additionally, POS tagging is associated with constituency parsing as it allows the parser to accurately place words in the appropriate constituency (Chang, 2016; Singh, 2018).

In Figure 4.3 the following keys are applied *S= Sentence*, *N =Noun*, *VP = Verb Phrase*, *NP = Noun Phrase*, *D = Determiner and V = Verb*.



Source: (Carnie, 2013)
Figure 4.3 – Constituency Parse Tree

By contrast, Dependency Parsers analyses a sentence as a set of pairwise word-to-word dependencies, with each dependency having a particular type reflecting its grammatical purpose. In other words, Dependency Parsing focuses on the relationship between words; namely, the subject and object as opposed to a Constituency Parse Tree, which breaks a sentence into sub-phrases. Consequently, in Dependency Parsing there are no verb phases *(VP)* or noun phrases *(NP)* (Chang, 2016; Neubig, n.d.). Consider the same sentence '*John hit the ball'* demonstrated in the Constituency Parse Tree for this example. Figure 4.4 illustrates the simplicity of dependency parsing.



Source: (Carnie, 2013)
Figure 4.4 – Dependency Parsing

Dependency Parsing can convey the subjects and objects of a verb, as well as which words are modifying (describing) the subject. This can help find precise answers to specific questions. In this example we want to know who or what hit the ball. Therefore, identifying the subject of the verb *'hit'* will help us determine who or what hit the ball. In this scenario, '*John'* is the subject of '*hit',* which has the object *'ball'* (Carnie, 2013; Neubig, n.d.)

A thorough understanding of the syntactic structure of words and sentences is important when constructing an NLP-based steganalysis that will allow a test of the research hypothesis. Furthermore, it will support the understanding of how different NLP algorithms generate parse trees. Parse trees are beneficial and employed in numerous NLP applications such as Grammar Checkers, Automatic Summarisation, Sentimental Analysis, and Semantic Analysis (Singh, 2017). Additionally, the implementation of parse trees can assist in detecting syntactic anomalies in a text corpus that might suggest evidence of linguistic steganography.

4.2.4 Semantic analysis

Semantic analysis determines the possible implication of a sentence by focusing on the interactions among word-level meanings within a particular sentence. However, semantic analysis alone cannot determine the meaning of a sentence but rather through the convergence of various linguistic concepts which include the morphology, syntax, lexical and semantic structures of a sentence (Khurana, et al., 2017; Liddy, 2001). The primary objective of semantic analysis is to abate the syntactic structures and offer the meaning of a word, finding synonyms, word sense disambiguation, translating between natural languages, and populating a base of knowledge (Rajani and Hanumanthappa, 2016).

It is important to acknowledge that certain words are polysemantic (i.e. that they have the same spelling but different meaning when placed within various context). When using a

natural language, certain words can be ambiguous as they can be interpreted in different ways − subject to the context of their occurrence and the domain of the text. Hence, lexical semantic ambiguity is a fundamental characteristic of any natural language and a challenge in natural language processing. For example, the word *'bank'* could represent a river bank or a physical building containing a financial institution (Agirre and Edmonds, 2006; Rajani and Hanumanthappa, 2016). To determine the suitability of a particular word within a sentence NLP applications implement a process known as 'Word Sense Disambiguation' (Agirre and Edmonds, 2006; Rajani and Hanumanthappa, 2016).

Word Sense Disambiguation (WSD) is defined as the ability to determine which meaning of a word is invoked by its implementation within a particular context. Lexical ambiguity, whether syntax or semantic, remains one of the fundamental challenges of natural language processing. Whilst POS taggers with a high level of accuracy can resolve syntactic ambiguity, the process of resolving semantic ambiguity is a lot more challenging but can be achieved through the implementation of various word sense disambiguation techniques such as ontological modelling, word frequency distribution and pragmatic knowledge of the domain of a document. These techniques are essential to this research study, particularly when trying to establish the appropriateness of a word or sentence within a text-based cover medium and are therefore discussed in greater detail in the later sections of this chapter (Agirre and Edmonds, 2006; Bird, Klein and Loper, 2009).

4.2.5 Pragmatic analysis

Pragmatics is a branch of linguistics that is concerned with the meaningful use of language within a situation and analyses the context instead of the actual content of a text to better understand it. Therefore, the objective of pragmatics is to adopt a more practical approach to a body of text rather than a syntactic or semantic one. Any NLP solution wanting to implement pragmatic analysis requires vast knowledge of the real world, that includes appropriate rules, intentions and relationships between the entities identified in a text. Hence, NLP systems that perform pragmatic analysis employ the use of knowledge bases, neural networks and inference models to construct the ontologies and metaphysics for entities identified in a text corpus (Nordquist , 2020; Jurafsky and Martin, 2008; Liddy, 2001).

Even though pragmatic analysis does not focus on the semantics of a text it can however be used to infer it. This is achieved through the establishment of ontology models which are used to '*complete the picture'*, by demonstrating how entities in a text relate to one another within the text itself or in the broader world. Ontologies provide a vital link to the real world and are now being recognised as important components of natural language processing. For example, assume that there is a large corpus of unstructured data; where do we begin? Without an ontology model an NLP parser would produce random interpretations having no real link to the world and lack complete meaning. Therefore, to perform pragmatic analysis, NLP systems must first establish an ontological model (Craig, 2012).

Pragmatic analysis from a computational perspective is extremely challenging and limited. This was demonstrated in the natural language understanding system named 'Relatus', developed by Gavan Duffy and John Mallery (Duffy, 2008). Relatus was able to convert English text into a network representation of nodes that were linked to one another based upon their similarities. In doing so, Relatus was able to rearrange these networks of nodes in various ways to infer different meanings. After parsing and representing a narrative, Relatus was able to respond to queries in English about the narrative's context. Despite these achievements Relatus was limited to a particular context and could not respond to queries beyond its context domain (Duffy, 2008).

In terms of this research study and most NLP systems, pragmatic analysis offers a more supportive rather than a central role. Generally, pragmatic analysis is implemented if any ambiguities arise at the syntactic and semantic levels of analysis where the context and purpose are considered for analysis. An example of which would be in the resolution of noun phrases found in constituency parsing (Chang, 2016).

4.3 Implementing NLP Techniques to Handle Linguistic Analysis

Now that the key processes to linguistic analysis have been presented and discussed, the different NLP computational methods or algorithms need to be examined that will assist in realising the research objective stated earlier. Additionally, it will help identify which NLP algorithms can be integrated into the development of the proposed steganalysis used to test

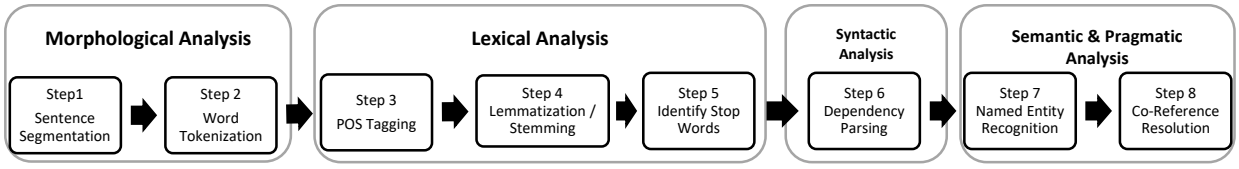
the research hypothesis. This is accomplished by implementing an NLP pipeline which is presented in the following sections (Khurana, et al., 2017).

4.3.1 Linguistic analysis through the implementation of an NLP pipeline

Computational processing of a natural language is an extremely challenging task as computers generally process structured data such as those in tables or databases. However, in the natural world people communicate in numerous ways and inflections resulting in large volumes of unstructured data making it extremely problematic for computational algorithms to process or extract any '*meaning'* from it. Consequently, the proposed steganalysis must initially transform and pre-process any unstructured corpora into something that an NLP algorithm would accept. This must be achieved prior to any feature extraction or the classification of stegospam from non-stegospam files (Fortney, 2017; Geitgey, 2018; Shaikh, 2018).

Consequently, this section will examine the different NLP pre-processing methods and algorithms required to transform unstructured corpora into a structured dataset that is acceptable for further NLP processing.

As with any major task, the strategy is to establish a step-by-step procedure to resolve the problem. The rationale in this approach is to breakdown the problem into more achievable subtasks, which in terms of this research study, is demonstrated in the NLP pipeline shown in Figure 4.5. The NLP pipeline encapsulates several machine learning algorithms or models to resolve each subtask independently (Geitgey, 2018). Figure 4.5 below illustrates the steps involved in the NLP Pipeline and the associated Linguistic Analysis phase.



Source: (Geitgey, 2018)

Figure 4.5 – Natural Language Processing pipeline with associated linguistic phases

- **Step 1: Sentence Segmentation.** In linguistics analysis, including NLP it is important to explicitly define what constitutes a word and a sentence as it supports our understanding during the segmentation or decomposition process. According to the Oxford Dictionary (2019), a word is defined as *"a string or group of letters that convey a sense"* and a sentence is defined as:

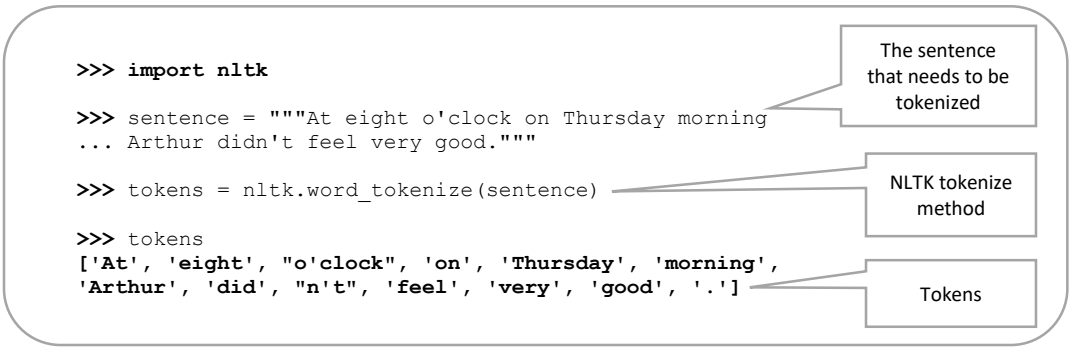
  > *a set of words that is complete in itself, typically containing a subject and predicate, conveying a statement, question, exclamation, or command, and consisting of a main clause and sometimes one or more subordinate clauses.*

Many natural languages, including English, contain several inherent ambiguities, hence much of the challenge of natural language processing involves resolving such ambiguities. Therefore, the first step in the NLP pipeline is to segment a text corpus into individual sentences. Subsequently, it can be assumed that each sentence is an isolated idea or concept that represents a linguistically meaningful unit as per the definition of a sentence. As illustrated in Figure 4.5, this step of the NLP pipeline is associated with the Morphological Analysis phase of Linguistic Analysis which deals with identifying the most basic unit of meaning. Sentence segmentation is an important step of NLP as the sentences identified in this step are fundamental units for further processing down the NLP pipeline, which includes tokenizing, POS tagging, normalisation and parsing (Geitgey, 2018; Palmer, et al., 2000).

The rationale for the implementation of sentence segmentation is that logically it is easier to construct a program or algorithm that can analyse a single sentence than to analyse an entire paragraph. Hence, developing a sentence segmentation algorithm can be as simple as separating sentences apart whenever it encounters a punctuation mark (Geitgey, 2018). Consequently, sentence segmentation has been implemented in the proposed steganalysis and is discussed in greater detail in Chapter 5 (Research Design and Methodology).

- **Step 2: Word Tokenization.** Once the text corpus has been segmented into sentences, the next step in the NLP pipeline is to segment each sentence into individual words, punctuation marks and symbols which are regarded as tokens. Word segmentation is a process also known as word disambiguation or tokenization. Word tokenization is a subtask of the morphological phase of the NLP pipeline and is an established process of Artificial Intelligence. Tokens are generally individual words that are used as input for further NLP processing. However, it is important to acknowledge that punctuation marks such as *'?'* or *'!'* are also treated as tokens (Bird, et al., 2009; Geitgey, 2018). Tokenization represents a form of pre-processing, an identification of the basic units to be processed further down the NLP pipeline. The sentence: *'Pretoria is the capital of the Republic of South Africa!'* can be seen to work with tokenization if it is further assumed that this sentence was extracted from a text corpus through sentence segmentation discussed in the previous section. Tokenizing this sentence would result in the tokens *'Pretoria'*, *'is'*, *'the'*, *'capital'*, *'of'*, *'the'*, *'Republic'*, *'of'*, *'South'*, *'Africa'*, *'!'* (Geitgey, 2018).

  In terms of linguistic analysis, a practitioner at this point would further strip a token to its lexical morpheme. Therefore, a token such as *'worked'* (regarded as a grammatical morpheme) will be stripped to its lexical morpheme *'work'*. Computationally, however, the NLP pipeline functions a little differently. During the process of tokenization, the token 'worked' remains in its grammatical sense as the purpose of word tokenization is to simply break down a sentence into more manageable pieces for further processing down the NLP pipeline including stripping a token to its lexical morpheme (Geitgey, 2018; Khurana, et al., 2017). Code Extract 4.1 – presents an extract of the Python code that implements the NLTK library used to tokenize the sentence – *'At eight o'clock on Thursday morning Arthur didn't feel very good'*. Further discussions on the code implementation is presented in Chapter 5 (Research Design and Methodology).



Source: (Bird, Klein and Loper, 2009)
Figure 4.6 - NLTK Tokenization method

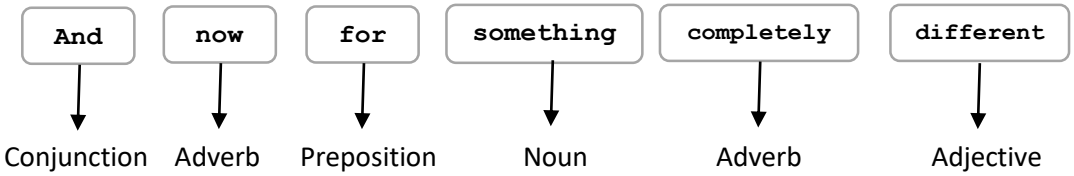
- **Step 3: Predicting Part-of-Speech (POS) Tags.** Next in the NLP pipeline is the prediction of the POS tags or labels for each token generated in the word segmentation process. In sum, POS tagging is a linguistic process that attempts to best identify the language class of each token such as nouns, verbs, adjectives or adverbs. The POS tags are also known as lexical tags, morphological classes or word classes, hence, it is associated with the Lexical Analysis phase of the NLP pipeline as illustrated in Figure 4.5 (Geitgey, 2018).

  Traditionally, POS annotation was handled manually which proved to be a laborious task. Fortunately, advancements in Machine Learning and Natural Language Processing has resulted in the development of several computational POS models or algorithms (Hasan, UzZaman and Khan, 2007). Computational POS tagging is achieved through the use of a POS method that enables an NLP library such as SpaCy or NLTK to make a POS prediction. The POS tagging can be either supervised or unsupervised. Supervised POS tagging requires pre-tagged training corpora to learn from and a probabilistic algorithm such as N-gram, Maximum Entropy Model[24] or Hidden Markov Model[25] to make its predications (further discussion on these prediction algorithms are presented in the footnote links).

[24] *https://mtlsites.mit.edu/Courses/6.050/2003/notes/chapter9.pdf*
[25] *https://medium.com/analytics-vidhya/hidden-markov-model-part-1-of-the-hmm-series-3f7fea28a08*

Unsupervised POS tagging does not require pre-tagged training corpora, instead it implements advanced computational algorithms like the Baum-Welch algorithm[26] to automatically construct POS tagsets (Hasan, et al., 2007). It is important to acknowledge that a POS model is based on statistical inferences and does not actually understand the context or ontology of a word; it simply predicts the POS tag based on the quality of the training corpora and predication algorithm (Geitgey, 2018; Hasan, UzZaman and Khan, 2007).

The POS tagging has been implemented in various tasks including information retrieval, parsing, information extraction, linguistic research, and as a preliminary step for more complicated NLP tasks including syntactic parsing, semantics analysis, and language translation (Hasan, UzZaman and Khan, 2007). Figure 4.7 provides an example for the tokenization and POS tagging of the sentence – *'And now for something completely different'*.



Source: (Bird, Klein and Loper, 2009)
Figure 4.7 – POS Tagging

The implementation of POS tagging in the proposed steganalysis is discussed in Chapter 5 (Research Design and Methodology).

- **Step 4: Token Normalization.** In English and most natural languages, words appear in different forms or inflections such as cars, car's or cars'. Reducing these words to its base form is extremely important, as computationally, the computer cannot infer that this is talking about the same concept. In NLP the process of finding the base form can be achieved by employing one of two methods, i.e. stemming or lemmatization (Geitgey, 2018). The objective of both stemming and lemmatization

[26] *https://medium.com/analytics-vidhya/baum-welch-algorithm-for-training-a-hidden-markov-model-part-2-of-the-hmm-series-d0e393b4fb86*

is to reduce the inflectional form and sometimes derivationally related form of a word to its common base form (Jabeen, 2018). For example, the tokens *'playing'*, *'played'*, or *'plays'* can be normalised to its common base form which is *'play'*.

Stemming is a little different to Lemmatization in the way that it produces the base form of tokens. Stemming employs a heuristic approach that simply removes or replaces the suffix or prefix of tokens to get to its root form (termed a stem) which can result in nonsensical words. For example, stemming a token such as *'fumbling'* would result in the token *'fumble'* (Fortney, 2017; Jabeen, 2018).

Lemmatization, in contrast, does things properly, by implementing a vocabulary and morphological analysis to return the base or vocabulary form of a token which is termed a 'lemma'. Using the same example in lemmatization would produce the base form of *'fumbling'* as *'fumble'*. Stemming and lemmatization are widely used in tagging systems, indexing, Web search results, and information retrieval. For example, searching for the word '*run'* on Google will yield *'runs'* and *'running'* as '*run'* is the stem for both these words. Subsequently, for stemming or lemmatization to function appropriately, it requires a decent lookup table of lemma forms of words based on their POS tags and custom-based rules to map such words (Geitgey, 2018; Jabeen, 2018).

In terms of this research study, the proposed steganalysis implements the Natural Language Tool Kit (NLTK)[27] library which offers both stemming and lemmatization. Based on the discussion presented, the proposed steganalysis implements lemmatization as opposed to stemming. The implementation and discussion of lemmatization as the preferred token normalisation approach for the proposed steganalysis is presented in Chapter 5 (Research Design and Methodology).

- **Step 5: Identifying Stop Words.** Next in the NLP pipeline is the identification and removal of stop words. Stop words are those words in a natural language that offer very little meaning and context to a document, such as *'is'*, *'a'*, *'the'*, etc. When

[27] *The Python NLTK library contains numerous methods that cater for various NLP tasks such as tokenizing, stemming, classification, parsing, tagging, and semantic reasoning* (Jabeen, 2018).

performing statistical analysis on text corpora stop words tend introduce a lot of *'noise'* and distraction since they appear far more frequently than other words. Hence, prior to further NLP or ML processing, stops words must be identified and filtered out.

The NLP libraries such as NLTK, SpaCy, Gensim[28] and TextBlob[29] contain a list of predefined stop words; however, these libraries do offer the ability to customise these built-in lists to suit the needs of an application (Geitgey, 2018; Teja, 2020).

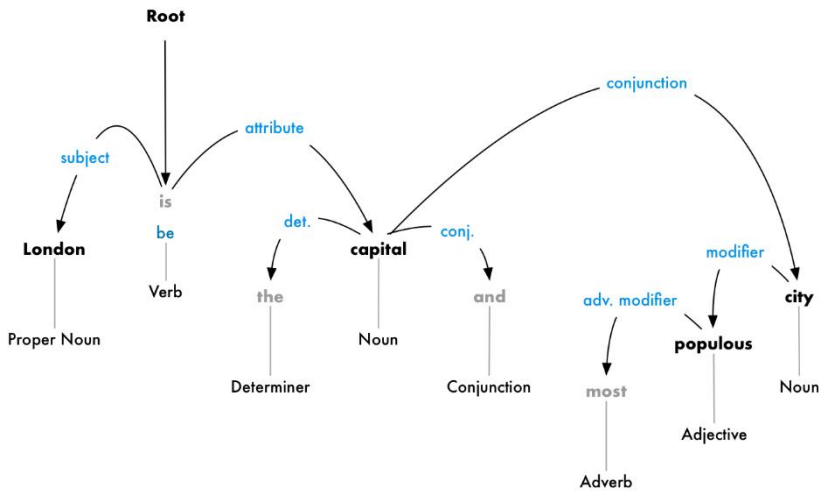
The removal of stop words assists in reducing the size of the training dataset, thus decreasing the time it takes to train a learning model without any impact on its accuracy. Additionally, the removal of stop words can improve the performance of an NLP/ML solution as there are fewer and only significant tokens remaining. Thus, the classification accuracy is improved. Conversely, improper stop word selection and removal can change the meaning of a sentence. For example, using the statement '*it is not good',* then removing the word *'not'* will result in the statement *'it is good',* which changes the entire meaning of the statement (Geitgey, 2018; Fortney, 2017). Consequently, to improve accuracy and performance, the proposed steganalysis will seek to identify and remove stop words from the input dataset. The implementation of this step is discussed in Chapter 5 (Research Design and Methodology).

- **Step 6: Dependency Parsing.** Up until this stage all the NLP methods discussed were related to the Lexical Analysis phase of the NLP pipeline. The next step in the pipeline is to ascertain the grammatical accuracy of each sentence, to achieve this we move into the Syntactic Phase which comprises of the Dependency or Syntactic Parsing process. During this process it can be discovered how the words of a sentence are related to one another by investigating its grammatical structure (known as its syntax) (Chang, 2016; Kadam, 2016).

[28] *Gensim is a Python library for topic modelling, document indexing and similarity retrieval for large corpora* (Rehurek, 2020).
[29] *TextBlob is a Python library for processing textual data* (Steven, 2020).

As mentioned in Section 4.2, syntactic analysis can be achieved through one of two approaches; namely, – Phase Structure or Dependency Parsing Trees. Owing to the popularity of Dependency Parsing employed in several PCFGs, this research study will focus on Dependency Parsing. Fundamentally, a Dependency Parser is a computational algorithm that translates a sentence into a Dependency Parsing Tree. A Dependency Parsing Tree is a structure that can be defined as a graph, with nodes (vertices), representing words, and Arcs, representing the syntactic dependencies or relations between them. These relationships communicate details about the dependency type[30] (e.g. Subject, Direct Object Complement, Punctuation etc.) (Cothenet, 2019).

The objective in this step is to develop a parse tree that assigns a parent word to each word in the sentence. A dependency parse tree includes a root which is the main verb in the sentence and a prediction of the type of relationships that exist between the root and other words of the sentence. An example of this is presented below in Figure 4.8 – Dependency parsing relationships for the sentence – '*London is the capital and most populous city':*



Source: (Geitgey, 2018)

Figure 4.8 – Dependency parsing relationships

In Figure 4.8 the word *'is'* represents the root verb of the sentence. This parse tree demonstrates that the subject of the sentence is the Proper Noun *'London'* and it

[30] *List of Universal Dependencies can be found at: https://universaldependencies.org/u/dep/index.html.*

has a *'be'* relationship with *'capital'*. Therefore, from this relationship something useful can be learned — *London is a capital!* Consequently, if the complete parse tree for the sentence (beyond what is shown) is followed, it can be discovered that London is the capital of the United Kingdom (Geitgey, 2018; Cothenet, 2019).

Discovering the syntactic accuracy of a sentence may prove to be an important feature for the detection of linguistic steganography. However, PCFGs employ several strategies including dependency parsing to generate syntactically accurate corpora such as the mimicry Spammimic (Chang and Clark, 2014) Hence, an alternative approach may be required for the proposed steganalysis to mine deeper and more meaningful features as discussed in Chapter 5 (Research Design and Methodology).

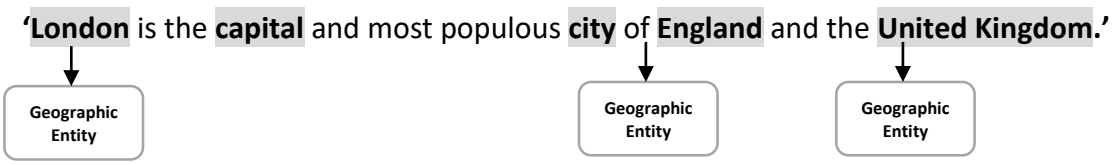
- **Named Entity Recognition (NER).** The final steps of the NLP pipeline relate to the Semantics and Pragmatics analysis phase of the NLP pipeline and looks to establish an ontological model of a corpus. Take the following sentence as the test corpus:

'London is the **capital** and most populous **city** of **England** and the **United Kingdom.'**

Subjecting this sentence to the steps and tasks specified in the NLP pipeline thus far, would result in the nouns highlighted. The identification of nouns is critical when establishing an ontology model that relates to the real-world. Some NLP applications prefer to group related nouns to establish some degree of context and to extract further information; however, this step is subjective and depends on the application's end goal (Geitgey, 2018).

Most of the tokens tagged as nouns, in the test corpus represent tangible things in the world such as *'London'*, *'England*' and *'United Kingdom'*, which represent actual geographical locations. Unfortunately, a computer cannot infer this, but if it could, a wealth of information could automatically be extracted. Therefore, this introduces an NLP procedure termed 'Named Entity Recognition' (NER), which is used to detect and map these nouns with real-world concepts that they represent (Geitgey, 2018).

Parsing each token through an NER tagging algorithm would result in the following:



To classify the nouns as geographical entities, NER algorithms do not merely perform a mere dictionary lookup; instead, they apply the context of how a noun is used within a sentence together with the implementation of a statistical model in order to predict the type of noun a token represents. A good NER algorithm is able to distinguish between '*Ivy'* the person and the plant '*Ivy'.* The NER systems are incorporated into our everyday lives to tag entities such as people, companies, products and events via the support of statistical models such as Support Vector Machines (SVM) (discussed in the next section) and Conditional Random Field[31] (CRF) - a discriminative directionless probabilistic graphical model that is used to translate known relationships between observations and construct reliable interpretations. The CRF is commonly implemented in parsing of sequential data such as those found in NLP tasks *(*Geitgey, 2018; Onal and Karagoz, 2015).

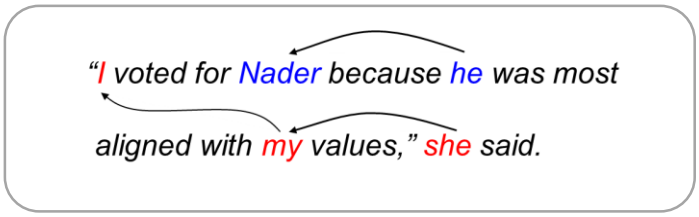
An NER is not without its limitations; one of the key reasons for the decline in recognition performance of NER algorithms is the informal and noisy nature of social media text. Social media text is ripe with spelling, punctuation, grammatical and capitalization errors. The NER algorithms fail to adapt to this new genre of text because they were originally designed for formal text and were based on features presented in well-formed text, which is certainly not the case on social media (Onal & Karagoz, 2015).

This research study will consider the NLP libraries SpaCy or NLTK for the implementation of NER. The identification of unique identifiers within a corpus may prove a vital feature in unveiling the signature of a steganographic tool and subsequently the stego-objects generated by it. Additionally, NER may be implemented in a scrubbing process, to remove '*unnecessary'* identifiers such as

[31] *https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776*

people's names, locations, monetary symbols, quantities etc. from the input corpora. In doing so, it may help to improve the performance and accuracy of proposed steganalysis.

- **Co-Reference Resolution.** Thus far, this section has presented and discussed the different NLP data pre-processing techniques associated with the corresponding linguistic analysis phases. This has enabled useful representations to be elicited from the sample corpora. However, the problem of pronouns such as *'he'*, *'she'* and *'it'* are still presented. Pronouns essentially represent shortcuts that are used as opposed to writing out the same name repetitively in each sentence. The problem persists because most NLP models cannot identify pronouns, as they are limited to processing a single sentence at a time. Consequently, NLP systems implement a process known as *co*-reference resolution which refers to the task of finding all words that refer to the same entity in a text corpus. Coreference resolution is an important step for various high level NLP tasks that involves natural language understanding such as auto-summarization, question answering, and information extraction (Geitgey, 2018). Figure 4.9 below, illustrates the use of pronouns*:*



Source: (StanfordNLP Group, 2018)
Figure 4.9 – Use of Pronouns

Co-referencing remains one of the more challenging tasks of the NLP pipeline. Combining several of the previously mentioned NLP techniques, including a parse tree and NER, should enable the mining of valuable information from a corpus but not necessarily at the level one would expect from a linguistic analysis perspective. Recent advancements in deep learning and neural networks have resulted in innovative approaches that are more accurate but not perfect. This research study will consider the implementation of the *neuralcoref* module which is based on the super-fast SpaCy parser for co-reference resolution (Clark and Manning, 2016;

Geitgey, 2018). The implementation of the *neuralcoref* for the proposed steganalysis is discussed in Chapter 5 (Research Design and Methodology).

4.4 Feature Extraction

The previous section presented and discussed the different NLP techniques employed in the pre-processing of raw, unstructured corpora into structured datasets that can be used for further NLP or ML tasks. To test the research hypothesis, a corpus containing a collection of stego and non-stegospam files needs to be built. This requires a pre-process of the corpus to extract tangible features from the pre-processed dataset and then to use the features to train a classifier which then enables it to classify stego from non-stegospam files. Consequently, the previous section discussed the pre-processing of a corpus, hence this section addresses the extraction of features used to train a classifier. This entire process is discussed in greater detail in the Chapter 5 (Research Design and Methodology).

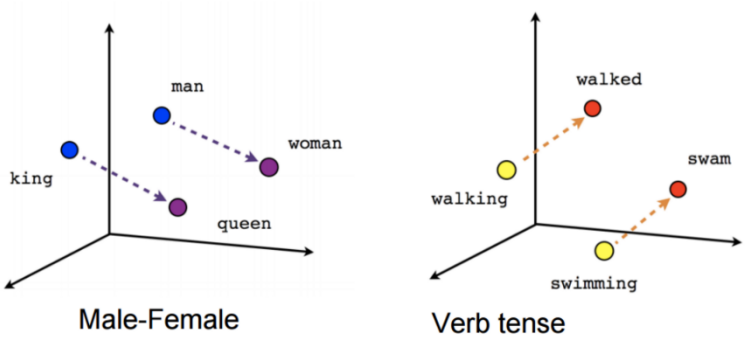
Feature extraction is a key process that functions within the context of machine learning and is used to evaluate the similarities between text corpora. Machine learning algorithms learn from a pre-defined set of features found within a training set with the aim of classifying a testing set. Thus, proper feature extraction improves the accuracy of learning models and reduces the dimensionality of the data by eliminating redundancies. Subsequently, improving training and inference speeds. Machine learning algorithms cannot work directly with raw text; hence, feature extraction techniques must be implemented to convert text into a vector of features required by learning or classification models (Kothari, 2020; Clark and Manning, 2016). There are several feature extraction models implemented in NLP which include Word2Vec, N-Gram[32], Fast*Text*[33] and Doc2Vec. However, for the purposes of this research study, focus will be on the discussion and implementation of the Word2Vec and Doc2Vec feature extraction models.

[32] *http://uc-r.github.io/creating-text-features*

[33] *https://towardsdatascience.com/fasttext-ea9009dba0e8*

4.4.1 Word2Vec

First introduced by Mikolov et al., in 2013, Word2Vec is a two-layer, shallow neural network that accepts text as its input and generates numerical vector representations for each token as its output. Word2Vec learns from a distributed representation of words found within a vectorspace to achieve improved performances in several natural language processing tasks. It achieves this by examining and grouping words with mathematical similarities into the same vectorspace. This is grounded on the hypothesis that words that occur in similar contexts (neighbouring words) tend to have similar meanings as illustrated in Figure 4.10 (Ali, 2019; Mikolov, et al., 2013; Nicholson, 2019).

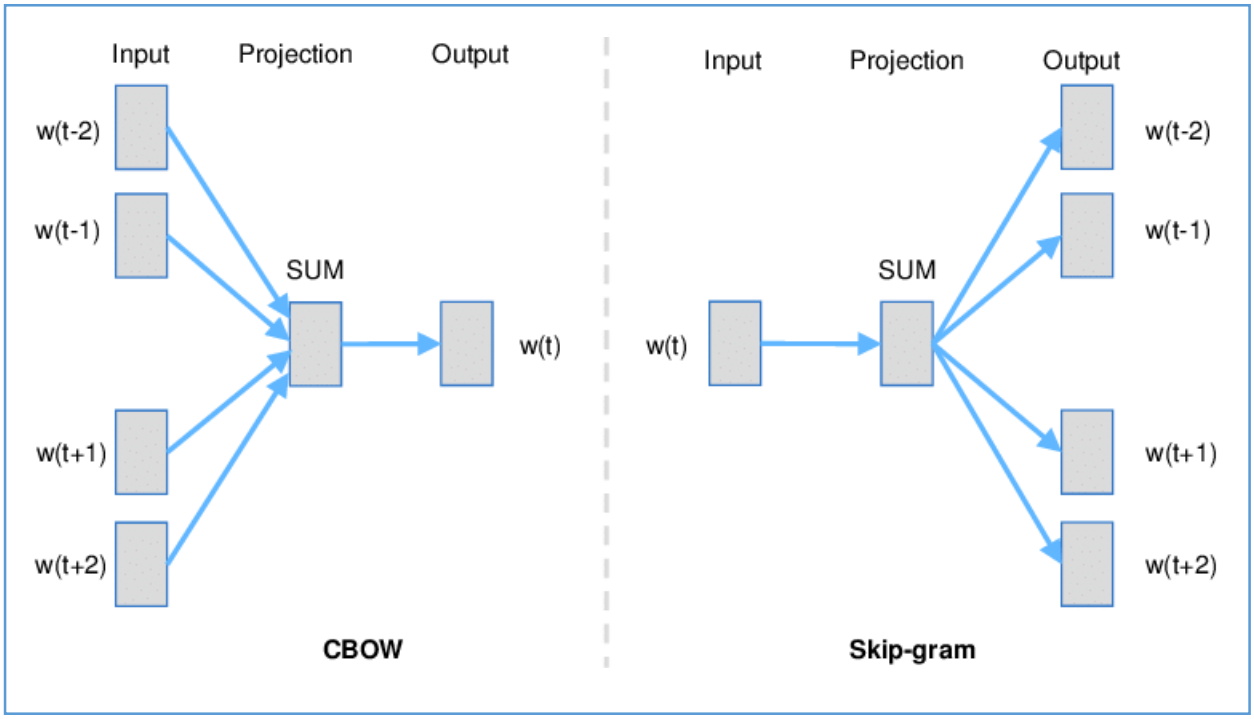


Source: (Kumar, 2019)
Figure 4.10 – Word2Vec – Vector Representation and Grouping

Neural word embeddings (the vectors used to represent words) calculated via Word2Vec have several encoded linguistic similarities and patterns (that are represented in a numerical form) can be fed into a classifier or deep-learning model to make predictions or simply queried to discover the relationships between words (Mikolov, et al., 2013). Consequently, Word2vec is analogous to an auto-encoder, encoding each token within a vectorspace; however, rather than training against the input words, Word2Vec trains words against those that neighbour them in the corpus (Nicholson, 2019).

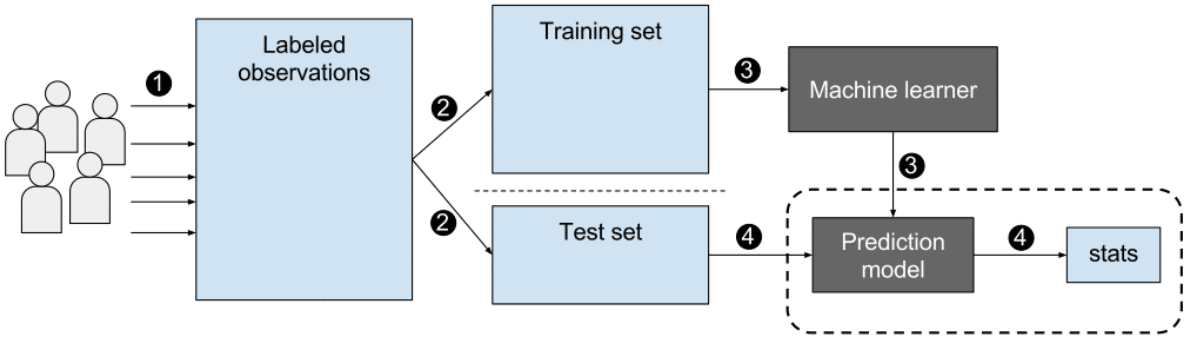
The Word2Vec can implement one of two model architectures to generate neural word embeddings; these include the Continuous-Bag-Of-Words (CBOW) or the Skip-gram model. In the CBOW model, the neighbouring context words are combined to predict the centroid word. The order of the nieghbouring words does not influence the prediction rating of CBOW. While the Skip-gram model uses the current word to predict the nearest surrounding context words. Skip-gram emphasises more on the neighbouring context words than on the distant

context words (Kumar, 2019; Mikolov, et al., 2013; Nicholson, 2019). Figure 4.11 illustrates the CBOW and Skip-Gram Models employed by Word2Vec.



Source: (Nicholson, 2019)
Figure 4.11 – Continuous-Bags-of-Words (CBOW) and Skip-Gram Model

The implementation of Word2Vec may be either supervised or unsupervised. In a supervised learning approach, the learning algorithm in this case Word2Vec is trained using a fully labelled dataset in order to extract deep and meaningful features. Fully labelled implies that each sample in the training set is labelled with the answer the learning algorithm should discover on its own (Mikolov, et al., 2013; Salian, 2018). However, in an unsupervised learning approach, Word2Vec is trained using an unlabelled training dataset. The training dataset is an assembly of samples without a specific label. The neural network then attempts to automatically find structure in the data by extracting useful features and analysing its structure to construct its feature set (Salian, 2018). Figure 4.12 illustrates the steps involved in a Supervised Learning Model.



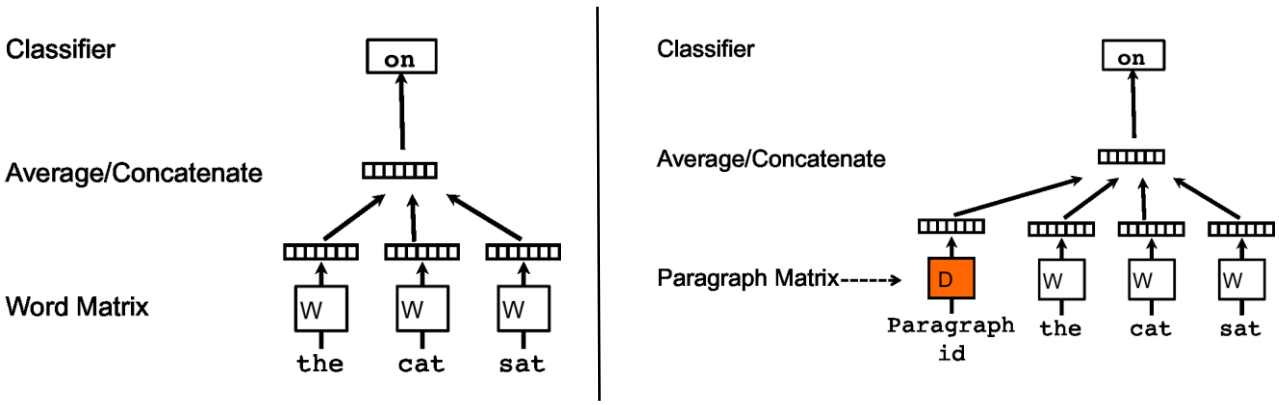
Source: (Salian, 2018)

Figure 4.12 – Steps involved in the Supervised Machine Learning Model.

4.4.2 Doc2Vec

The Doc2Vec is an unsupervised, paragraph-vector learning model that learns continuous distributed vector representations for variable-length text ranging from single sentences or phrases to large documents. The Doc2Vec model was first introduced by Le and Mikolov in 2014, which was built on the research paper '*Distributed representations of words and phrases and their compositionality*' by Mikolov et al. 2013, essentially Word2Vec. Unlike Word2Vec, the Doc2Vec is capable of constructing vector representations of input sentences of variable length and does not require task-specific tuning of the word weights nor rely on the parse trees (Le & Mikolov, 2014).

The Doc2vec model is quite simple as it is based on the Word2Vec model, but includes an additional vector – '*Paragraph ID*' to the algorithm. A comparison between the Word2Vec and Doc2Vec architectures is illustrated in Figure 4.13 below where the Paragraph ID can be seen added to the input to construct the Paragraph Matrix.



Source: (Le & Mikolov, 2014)

Figure 4.13 – Doc2Vec Architecture

As with the Word2Vec, the Doc2Vec may be implemented using an unsupervised or supervised approach. The ability to extract unbiased, deep relations or features from sentences and documents from a text corpus is a strong motivator for the implementation of Word2Vec and Doc2Vec in the proposed steganalysis. Additionally, these models promote the concept of generalisation, in that one could input stego-objects generated by any linguistic steganographic tool into the proposed steganalysis and it would be able to extract deep features and use them to classify or detect the stego-object. Consequently, the consideration and implementation of Word2Vec and/or Doc2Vec is discussed in Chapter 5 of this research study. In the next section the classification models used to learn the extracted features to make predictions thereof are discussed.

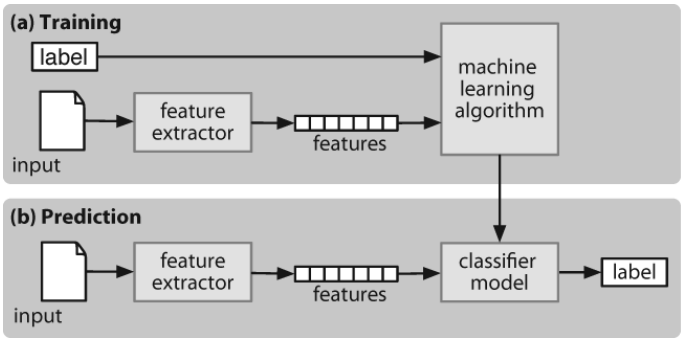
4.5 Classification Models

Classification is the process of assigning the correct class label for a given text where each data point (i.e. a file or sample*)* within the text corpus is handled in isolation of one another. Text classification is one of the essential tasks in natural language processing or machine learning with extensive applications in sentiment analysis, topic labeling and spam detection. Classification models such as Naïve Bayes, Support Vector Machine (SVM) and Random Decision Forest (RDF) accept pre-tagged or labelled training sets assembled using different feature extraction models such as Word2Vec or Doc2Vec to classify text or make a prediction (Bird, Klein and Loper, 2009; Geitgey, 2018; Giyanani and Desai, 2014).

Similar to the feature extraction models discussed in the previous section, there are two primary classification approaches i.e. supervised and unsupervised. A supervised classifier is one that is implemented using a training set comprising of predefined labels for each data point within the training set. By contrast, unsupervised classifiers are implemented using training sets that do not have predefined labels for each data point, instead they take advantage of neural and collaborative networks to classify the data (Bird, Klein and Loper, 2009; Geitgey, 2018).

With text classification, the algorithm does not concern itself with the language, pragmatic or semantic structure of a text corpus but rather the statistical relationship between the input phrases and outputs. Hence, a classifier is able ascertain what words or phrases mean in the

context of where they appear and how frequently they contribute to a particular output. Consequently, an NLP pipeline trained to handle the English language will fail if the targeted test set is in Afrikaans or isiZulu (Bird, Klein and Loper, 2009; Geitgey, 2018). Figure 4.14 illustrates the process for a supervised classifier.



Source: (Bird, Klein and Loper, 2009)
Figure 4.14 – Supervised Classification Process.

In Figure 4.14, during the training phase (a), a feature extractor or mining module such as Word2Vec is implemented to convert the input training dataset to a feature set. These feature sets capture the fundamental information about each input that can be used to classify it. Pairs of feature sets and tags are then fed into a machine learning algorithm to generate a model. During the prediction phase (b), the same feature extractor is implemented to convert testing dataset into feature sets. These feature sets are then passed into the model to make a prediction (Bird, Klein and Loper, 2009).

There are several methods to automatic NLP text classification which are categorised into one of the following two types of systems:

- **Rule-based system.** This method classifies text into groups through the implementation of predefined linguistic rules. These rules instruct the system to implement semantically appropriate elements of a text to recognize applicable categories based on its content. Each rule is composed of a pattern and a predicted category (Bird, Klein and Loper, 2009; Geitgey, 2018)
- **Machine learning-based system.** This method learns to classify text based on previous observations by using pre-labelled samples as its training set. Machine learning algorithms can learn the different associations between pieces of text and

that a particular output (i.e. label) is expected for a particular output (i.e. text) (Bird, Klein and Loper, 2009; Geitgey, 2018).

In relation to the classification models elected for the proposed steganalysis presented in Section 5.4 of Chapter 5, this section will discuss the machine learning based classification models - Naïve Bayes and the Support Vector Machine (SVM). Additionally, these models improve the rigour and reliability of the proposed steganalysis which is aligned to the positivistic research paradigm adopted by this study.

4.5.1 Naïve Bayes classifier

The Naïve Bayes' classifier is a probabilistic machine-learning model comprising a group of statistical algorithms implemented widely for the classification and analysis of text in several NLP/ML applications including spam filtering (Jøsang, 2016; Rusland, et al., 2017).

One of the algorithms of that group is the Multinomial Naïve Bayes (MNB) algorithm which achieves excellent results even when the input dataset is limited and computational resources are scarce. Naïve Bayes is based on the Bayes Theorem which enables the calculation of the conditional probabilities of the occurrence of events (Patel, 2019; Rusland, et al., 2017).

The fundamental difference between the Bayes' and the Naïve Bayes' algorithm is that Bayes' uses a sentence or phrase as a known condition such as, '*you have won a prize'* to determine its probability. Therefore, the probability P can be defined for a condition *{Spam}* in Bayes' as: *P{Spam|you have won a prize}*. Whilst Naïve Bayes is a lot more granular in that is assumes that each word is independent of the other and then calculates the probability of each word, and then finally multiplies the possibilities together, therefore, it is: *P{Spam|have} x P{Spam|won} x P{Spam|a} x P{Spam|prize}*. This is where the Naïve Bayes differs: in that it does not assume any relationship or dependencies between the words of a text corpus (Jøsang, 2016; Stecanella, 2019). Figure 4.15 illustrates the Naïve Bayes Algorithm.

$$P(\mathrm{A}|\mathrm{B}) = \frac{P(\mathrm{B}|A) \cdot P(A)}{P(B)}$$

Source: (Sayad, 2010; Jøsang, 2016)
Figure 4.15 – Naïve Bayes Algorithm

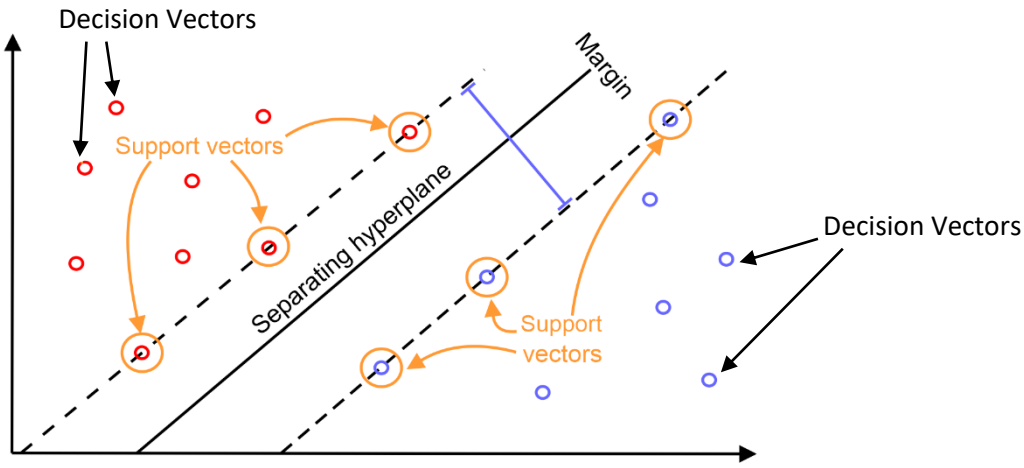
Based on the Naïve Bayes algorithm illustrated in Figure 4.15, the probability of A, if B is true, is equal to the probability of B if A is true, multiplied by the probability of A being true, divided by the probability of B being true. This implies that any vector that represents a text must contain information about the statistical probabilities of certain words within the text for a predefined label, such that the algorithm can compute the probability of that text belonging to that label (Jøsang, 2016; Rusland, et al., 2017).

The advantages of the Naive Bayes classifier are that (a) it works collectively for continuous and discrete attributes within the same dataset, (b) It converges rapidly to a solution with a smaller dataset and also can learn nonlinear concepts, and (c) It has a complexity of $X(n)$ (where n is total number of observations). The shortcoming of this classifier is that it implements a strong 'independence assumption' amongst features which in a real world problem may not be appropriate (Vyas, Prajapati and Gadhwal, 2015).

4.5.2 Support Vector Machine (SVM)

The SVM is supervised machine-learning algorithm that can be implemented for both classification and regression tasks. Like Naïve Bayes, SVM does not require extensive training to generate accurate results. However, SVM does require additional computational resources but produces faster and more accurate results than Naïve Bayes as demonstrated in the empirical evidence gathered and presented in Chapter 6 (Fletcher, 2009; Pupale, 2018; Vyas, Prajapati and Gadhwal, 2015).

The SVM is simple, the algorithm develops a line or 'hyperplane' to separate the vectors(tags) into different classes or labels. Support vectors are those vectors that are closet to the hyperplane and are consider the critical elements of the dataset. Consequently, SVM attempts to make a decision on vectors where the separation between the two classes is at its widest, known as 'Decision Vectors' (Fletcher, 2009; Pupale, 2018). This is illustrated in Figure 4.16 below.



Source: (Fletcher, 2009)
Figure 4.16 – Support Vector Machine (SVM)

The implementation of Naïve Bayes and the Support Vector Machine classification algorithms are discussed in Chapter 5 (Research Design and Methodology).

4.6 Natural Language Processing Review

This chapter addresses the research objective OB3: **Identify and describe various Natural Language Processing methods or algorithms that may be considered for the proposed steganalysis**

This is addressed by presenting and discussing various natural language methods and algorithms that may be considered for the design and development of the proposed linguistic steganalysis. These include key processes such as sentence and word segmentation, lemmatization, dependency parsing, feature extraction and classification. The challenge was to establish the research design and methodology that will be implemented to test the research hypothesis. Consequently, the design and development of the proposed linguistic steganalysis is discussed in greater detail in the next chapter.

5

Research Design and Methodology

This chapter presents the research design and methodology implemented to test the research hypothesis and to address the research objective - ***Identify and describe various natural language processing (NLP) methods or algorithms that may be considered for the proposed steganalysis.***

Introduction

The rationale behind the problem statement, research questions and objectives outlined in Chapter 1 were discussed in the preceding chapters. In Chapter 2, it was established that despite numerous developments in the field of spam filtering, spam email remains as pervasive as ever (Klensin, 2008). The innocuity associated with spam email has positioned it as a favorable carrier for numerous forms of covert attacks including steganographic attacks. This was emphasised in a case presented in Chapter 2, where a husband covertly conspired with a murderer via the use of a linguistic steganographic tool – Spammimic, which is freely available on the internet, to arrange for the murder of his wife (Yu, 2015). As this research steps forward from the literature review, it is essential to draw a strong sense of the characteristics associated with a preferred steganographic approach as those mentioned in section 3.3 (Properties to consider when electing a steganographic approach) namely - imperceptibility, robustness and capacity. These factors are key considerations when implementing a covert attack and will be referenced throughout this chapter to argue the decisions taken (Abdul-mahdi, Yahya and Ahmad, 2013).

This chapter outlines the research design and methodology implemented to test the research hypothesis and addresses the research objective stated above. Additionally, it will support the discussion of the proposed steganalysis through the construction and implementation of an NLP-based prototype and substantiate the motivation for choices taken as part of the research design.

This chapter starts with an introduction into the research that will be conducted and provides information on why certain decisions were taken. The research process and design are then discussed and thereafter the research methodology is presented. The limitations of the research and other related studies in this field are then presented. Finally, concluding remarks are provided.

The structure of this chapter is as follows:

Section 5.1 - Introduction – provides an introduction into this chapter.

Section 5.2 - Research Overview – provides an overview of the different research strategies.

Section 5.3 - Research Design - describes the research design adopted and argues the rationale for the research methodology adopted.

Section 5.4 - Research Methodology – Provides a detailed discussion of the research method used. It provides the details of the research environment, what data has been used and how it has been collected, the preprocessing and analysis of the research data, the NLP and classification algorithms used and finally, the evaluation of the research.

Section 5.5 - Identifies and discusses the limitations of the research.

Section 5.6 - Discusses related research in the field of linguistic steganalysis.

Section 5.7 Provides concluding remarks on the research methodology.

5.1 Background

A systematic, thorough process is required to evaluate the research hypothesis and determine if the proposed steganalysis can efficaciously detect or classify stegospam files. To answer the research questions and realise the objectives, the research methodology elected must be appropriate with respect to the research conducted, and clearly describe the process adopted to develop, implement, and evaluate the efficacy of the proposed NLP-based steganalysis.

Gounden (2012), defines research methodology as *"a systematic way to solve a problem*". Whilst, Marczyk, DeMatteo and Festinger (2005) describe it as a *"methodological and systematic approach to the acquisition of new knowledge"* which highlights the difference of how a scientist and non-scientist acquire new knowledge. Consequently, by conducting research within the field of information systems, this research study must conform to a well-structured and methodological approach. Rather than relying on casual observations and an informal approach to the analysis of stegospam, this research will, therefore, be steered using a controlled and methodical process. In addition, it is important to accept that the accumulation of scientific knowledge is not based on the opinions, feelings, or intuition of the researcher. Instead, it is grounded on data that is reliably obtained within the context of a carefully designed research study and the empirical evidence recorded (Marczyk, DeMatteo, and Festinger, 2005).

The rationale behind this research is grounded on the exploitation of spam email as a cover medium to conceal secret messages in plaintext format using linguistic steganography. The motivation behind the use of spam as a stego-object (referred to as stegospam) is detailed in the earlier chapters. However, the central motivation is based on the general supposition that spam email is innocuous; an assumption which attackers exploit to realise their objectives. The focus of this study is limited to stegospam files that are generated using linguistic steganography. This can be achieved through the use of a probabilistic context-free grammar *(PCFG)* tool such as Spammimic or Texto. Additionally, the acronym PCFG, is used to represent any software tool that can be used to auto-generate context-free stego-text by means of linguistic steganography.

The purpose of this research study is to test the hypothesis through the implementation of a prototype constructed using natural language processing methods and software libraries.

For the selection of an appropriate research methodology several factors were considered, these included the research objectives, the research methods adopted by similar studies within the same field, the underlying research philosophy, the composition of the input dataset that will be used to train, test and evaluate the constructed prototype, the NLP methods to be implemented, dependent and independent variables, the data analysis approach and the research questions.

The rest of this chapter discusses the research overview, research design, the underlying research philosophy, the research methodology and its limitations, and finally the evaluation of the empirical data produced.

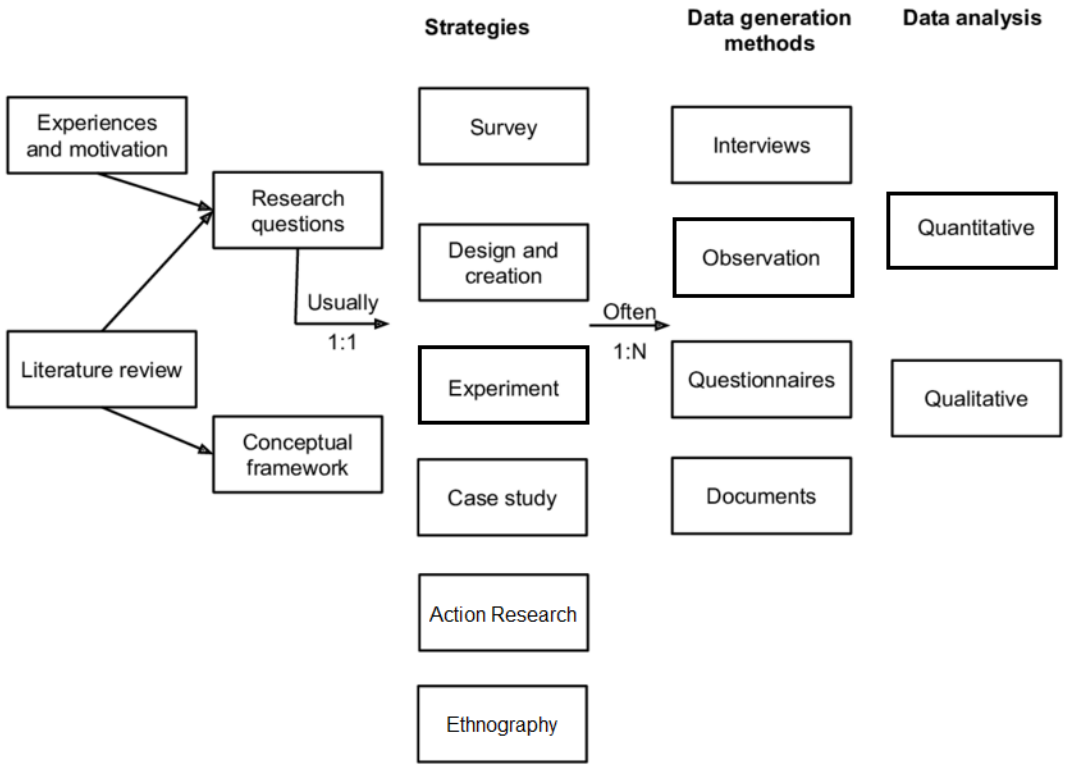
5.2 Research Overview

To appreciate the rationale for the election of a research method, an overview of the different forms of research methodologies is presented in this section. Each method is aptly suited to a specific condition or situation. To elect the most appropriate method, each method needs to be considered and evaluated on its merits pertinent to this study.

Research can be classified in a number of ways; one such approach is based on the type of data that is generated, which is either empirical or non-empirical. An empirical research approach is defined by Kazdin (2003), as *"an evidence-based approach that relies on direct observation and experimentation in the acquisition of new knowledge".* Furthermore, empirical methods are not only those that depend on observations, but may include surveys, questionnaires and experiments as shown in Figure 5.1. Consequently, the quality of this research is hinged on the accuracy of the observations and the degree to which the observed results can be generalised to other cases or studies (Marczyk, DeMatteo and Festinger, 2005; Olivier, 2009).

The empirical approach is a quantitative and positivistic one, which infers that the social world is perceived as governed by laws that render it predictable. Originally, empirical methods have

been associated with the use of quantitative measures (e.g. content analyses, surveys, experiments) and the collection and analysis of primary data. Nowadays, secondary analysis and qualitative research may also be considered empirical. It may be deemed credible to categorize qualitative research as empirical, only to the extent that researchers offer sufficient information that allows for the reproduction of their findings (Dan, 2018). Figure 5.1 below (Model of the Research Process) illustrates the research process and the various empirical and non-empirical research strategies.



Source: (Oates, 2006)
Figure 5.1 – Model of the Research Process

According to Oats (2006), there are a number of strategies or methods that may be used to design and conduct a research study. As with any research study the most appropriate method must be adopted to ensure that it generates the appropriate data to adequately test the hypothesis. The following subsection briefly discusses the different research strategies as depicted in Figure 5.1:

- ***Survey.*** An elicitation technique that allows a researcher to gather data from a prescribed and diverse population of people or events in a uniform and methodical manner. Deductions or inferences are then drawn through careful observation of the

patterns within the data collected. These patterns can then be generalised and applied to a much larger population than the one tested. Surveys do not only use questionnaires for data generation but may use techniques such as interviews, observations and document analysis (Oates, 2006). Whilst surveys are well-suited for demonstrating the correlation between characteristics, they are not intended to demonstrate causation (Olivier, 2009) and may not be a suitable strategy for this research which intends to demonstrate the cause and effect approach by implementing NLP techniques in the constructed steganalysis.

- ***Design and Creation.*** This research strategy is commonly associated with Information Systems (IS) research projects and involves the development of a prototype and the use of various data generation techniques such as interviews, observations, questionnaires and document analysis (Oates, 2006). The working prototype is usually expected to undergo real-world evaluation. Whilst it may appear superior to assess the prototype within a real-world context as opposed to a controlled environment, developers and students alike tend to deliberately ignore the risky or uncertain aspects of a project and only focus on a very specific problem with very little to no control of the environmental factors making it difficult to generalise (Oates, 2006; Goundar, 2012). Alternatively, a researcher could investigate the variables and focus on the risky aspects of a project in the hope that it may lead to new knowledge. Once achieved, the research is generalised and can be applied to other situations (Oates, 2006; Olivier, 2009). Although, this approach may to some extent appear appropriate, this research study does not envisage the implementation of a working prototype within the context of a real-world environment, nor will it use data generation techniques such as questionnaires or interviews. Further, certain variables will be measured and tested within a controlled environment using a cause and effect approach to prove or disprove the hypothesis. Consequently, this may not be the most appropriate method.
- ***Experiment.*** In academic research, an experiment is a strategy that investigates cause and effect relationships, seeking to prove or disprove a causal link between a factor and an observed outcome (Goundar, 2012). This methodology is often associated with research in the physical sciences and is at the core of the scientific method and

positivism. An experiment is designed to prove or disprove a hypothesis. All factors that might affect the results are excluded from the study, apart from those that is thought to cause a particular outcome. The experiment is performed and careful analysis and all observations are recorded. If the researcher is assured that no other factor(s) could have caused the observed results, then the hypothesis that factor A causes outcome B is proven (Marczyk, DeMatteo and Festinger, 2005; Oates, 2006; Olivier, 2009). Owing to the nature and the positivist epistemological paradigm of this study, the experimental methodology may be a suitable option.

- ***Case Study***. This approach produces rich, detailed information about a specific phenomenon when compared to other methodologies. However, one of the major challenges of this approach is to obtain knowledge that is useful and not just interesting facts about the phenomenon being studied. The data generated may be quantitative or qualitative and is subjected to the approach of the researcher (Olivier, 2009). A case study is characterised by the following factors:
    - its focus is on depth rather than breadth;
    - it is an examination of the instance or entity within its natural setting;
    - it is a holistic study that focuses on the complexity of relationships; and
    - It uses multiple sources and methods to gather or generate data.

  Case studies can be exploratory, descriptive and explanatory with each type producing detailed analysis on the phenomenon being studied. Whilst many may criticise case studies for generating knowledge that only applies to a particular case under study, it is possible to generate broader generalisations, only if they are relevant beyond the case itself. Generalisation is further achieved when multiple cases are used to proof more general results or when the case is typical to other cases. Unlike like other research approaches the underlying research philosophy may encapsulate positivism, interpretivism and critical thinking (Oates, 2006; Olivier, 2009).
- ***Action Research***. While action research represents a twenty-first century embodiment of practical philosophy, nevertheless, it differs in numerous critical respects. For example, in practical philosophy an understanding of the distinctive nature of practice is allowed to determine the kind of 'science' appropriate to its development; however, action research is developed in response to the need for a new social scientific

research paradigm that would eradicate the gap between theory and practise (Carr, 2006). It is generally used by researchers who wish to investigate and improve their own working practices. For example, educators might research strategies for maintaining discipline in the classroom and examine which seem to be the most effective strategy (Carr, 2006; Oates, 2006). Academics carrying out action research into a situation would be expected to collaborate with practitioners within that setting. Further, their role might change from experts to facilitators to enable practitioners to do their own research. This approach to research in information systems is not always apparent. Action research is characterised by the following factors:

- an iterative cycle of plan-act-reflect;
- an emphasis on change, collaboration with practitioners;
- multiple data generation methods; and
- a concentration on practical issues (Oates, 2006).

It is imperative not to make broad generalisations from just one action research study that might have unique features not found in other situations, as in case studies (Oates, 2006; Olivier, 2009). The research methods used by Action Research are rather qualitative than quantitative and is subjected to the perceptions of the researcher and the practitioners (Carr, 2006; Oates, 2006). The underlying research philosophy for this approach is more centred on interpretivism as opposed to positivism (Carr, 2006; Oates, 2006)

- ***Ethnography.*** Oates (2006) defines ethnography as a description of peoples or cultures. It refers to the study of users through direct observation in their natural environment rather than observations in a lab (Young, 2019). Ethnography is a qualitative research method that is used to gain insights into how users interact with entities within their natural environment. Ethnographic researchers gather and record data about the culture being studied, reflect on the process of how they came to understand the culture, acknowledge how they might have impacted on the people of that culture, link what they have learnt from previous literature, and then write up the process and findings as academic articles (Oates, 2006). Ethnography data generation methods, therefore, encapsulate direct observation, document analysis, video

recordings, photography and artefact analysis (Young, 2019). Since ethnography produces rich, detailed information about the culture of an entity, the information is referred to as 'stand-alone' and lacks broader implications or generalisation (Oates, 2006).

The preceding discussion has provided an overview of an empirical approach to academic research, the following paragraphs, however, discuss the non-empirical approach to research and the generation of data.

Non-empirical research endeavours to address questions that are not quantifiable in the physical world, and focuses on scientific concepts and theories. This type of research also generates qualitative data such as philosophical analysis, conceptual analysis, theory building and literature reviews (Mouton, 2004). Researchers employing non-empirical methods, consider that data generation methods such as reflection, personal observation and authority or experience are as tangible for knowledge acquisition as empirical or quantifiable data (Dan, 2018). However, this contradicts the scientific method of generating verifiable, empirical evidence (Oates, 2006).

Non-empirical methods can be subdivided into two categories. In the first category, the methods intended to review the progress in a certain field of research (e.g. literature review) can be found. In the second category, there are non-empirical methods that draw on personal observations, reflection on events, and/or the authority or experience of the researcher which are synonymous with the Case Study, Action Research, and Ethnography research methods as illustrated in Figure 5.1 (Dan, 2018; Oates, 2006).

In addition to empirical and non-empirical methods, there are two data analysis approaches to research: quantitative or qualitative. Quantitative research is based on a positivist paradigm, which implies that observations and measurements must be conducted in an impartial manner by means of data or evidence grounded on numerical values. On the other hand, qualitative research is based on the interpretative or critical thinking paradigms. In the qualitative approach, data is generated through the experience of an event or behaviour and includes non-numerical evidence such as images, audio, video etc. (Oates, 2006; Olivier, 2009).

Data analysis allows the researcher to identify patterns in the data generated and draw conclusions thereof (McNabb, 2010). In quantitative data analysis a broad range of established mathematical and statistical analysis techniques are also employed. However, in qualitative data, it is not so straightforward; there are no established mathematical or statistical analysis techniques, therefore, data analysis is reliant on the ability of the researcher to identify patterns and themes in the data, yielding an interpretative analysis as opposed to a positivist one (Oates, 2006).

This research study falls within the realm of Information Systems (IS) and employs the positivism philosophical paradigm, which is one of the scientific research philosophies. The fundamental objective of the scientific method is to discover universal laws, patterns and regularities and this is largely achieved through experimentation (Oates, 2006). Typically, researchers conduct experiments to search for evidence using a cause and effect method. They establish or propose a hypothesis that A causes B, and then search for evidence derived from carefully constructed experiments that confirm or refute the hypothesis (Oates, 2006; Olivier, 2009).

Experiments used extensively in Information Systems (IS) research are characterised by:

- precise and detailed observations of a factor;
- manipulation of independent variables; and
- re-observation or re-measurement of the factor to identify any changes.

These characteristics are aligned with the basic scientific method principles: reductionism, repeatability, and refutation which allow for the construction of knowledge through an iterative cycle (Oates, 2006).

Experimental research, however, is generally conducted to:

- discover something interesting (also referred to as an exploratory experiment);
- test a theory; or
- prove a theory (Olivier, 2009).

Given the hypothesis, objectives and nature of this study, the experimental research methodology was adopted as the preferred research method, thereby allowing this study to follow a structured and methodical approach to prove or refute the hypothesis through a series of controlled experiments to generate quantifiable empirical evidence. This approach conforms to the scientific method, the positivistic research paradigm and the generalisation of the proposed solution.

In this section the different research methods have been discussed, the next section presents the arguments and discussion related to the research design for this study.

5.3 Research Design

In the previous section, an overview of the different research methodologies was presented including the research method elected by this study. This section provides a discussion on the research design model adopted as illustrated in Figure 5.2 below.

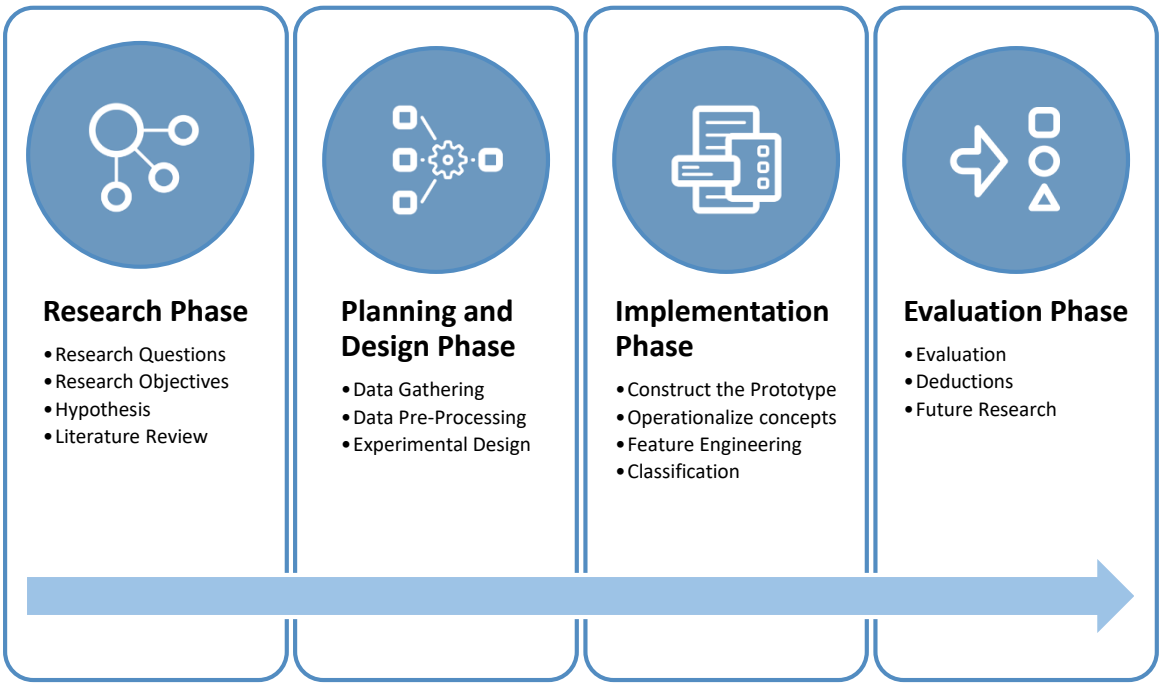


Figure 5.2 – Overview of the Research Design Model

The structure of the research design model is distinguished by four distinct phases with each phase containing key tasks. The first phase of the model, i.e. the Research phase, was addressed in Chapter 1 where the research questions, objectives and hypothesis were

presented. This was followed by the literature review, which discussed the motivation and the viability of spam as an ideal carrier for steganographic attacks, the conceptualisation of the different types and characteristics of steganography. Natural Language Processing (NLP) methods and procedures were presented and discussed in Chapters 2, 3 and 4 respectively.

The second phase of the research design model (as illustrated in Figure 5.2 above) is the Planning and Design phase, which consists of the data gathering, data pre-processing, and experimental design tasks. In the previous section, it was concluded that the experimental research method would be the preferred methodology for this study. Experimental research employs a quantitative approach, which encompasses the gathering of data so that it can be quantified and subjected to statistical analysis to support or refute the hypothesis. This approach is ideally suited to this study as quantitative research maintains the assumption of an empiricist paradigm (Creswell, 1994). The research method itself is independent of the researcher, hence creating meaning through objectivity discovered from the collected data (Williams, 2007). The text corpus gathered for this study was used to construct the training and testing sets, which were parsed into an NLP constructed prototype – i.e. a simplified version of a program or system (Olivier, 2009). Whilst a valuable tool in experimentation (the design and development of the prototype) does not constitute research per se, it is merely developed to test the hypothesis and establish proof of concept (Olivier, 2009).

To test the hypothesis, it is imperative that the input dataset is composed of plaintext files. This is an essential requirement for all NLP tasks (Bird, Klein and Loper, 2009). The next step in the Planning and Design Phase is the data pre-processing step. This step includes the analysis and nomination of relevant data scrubbing and pre-processing methods required to prepare the unstructured, raw corpora into a single well-structured dataset, demanded by the prototype (Bird, Klein and Loper, 2009). Data preprocessing is explained in greater detail in the following sections.

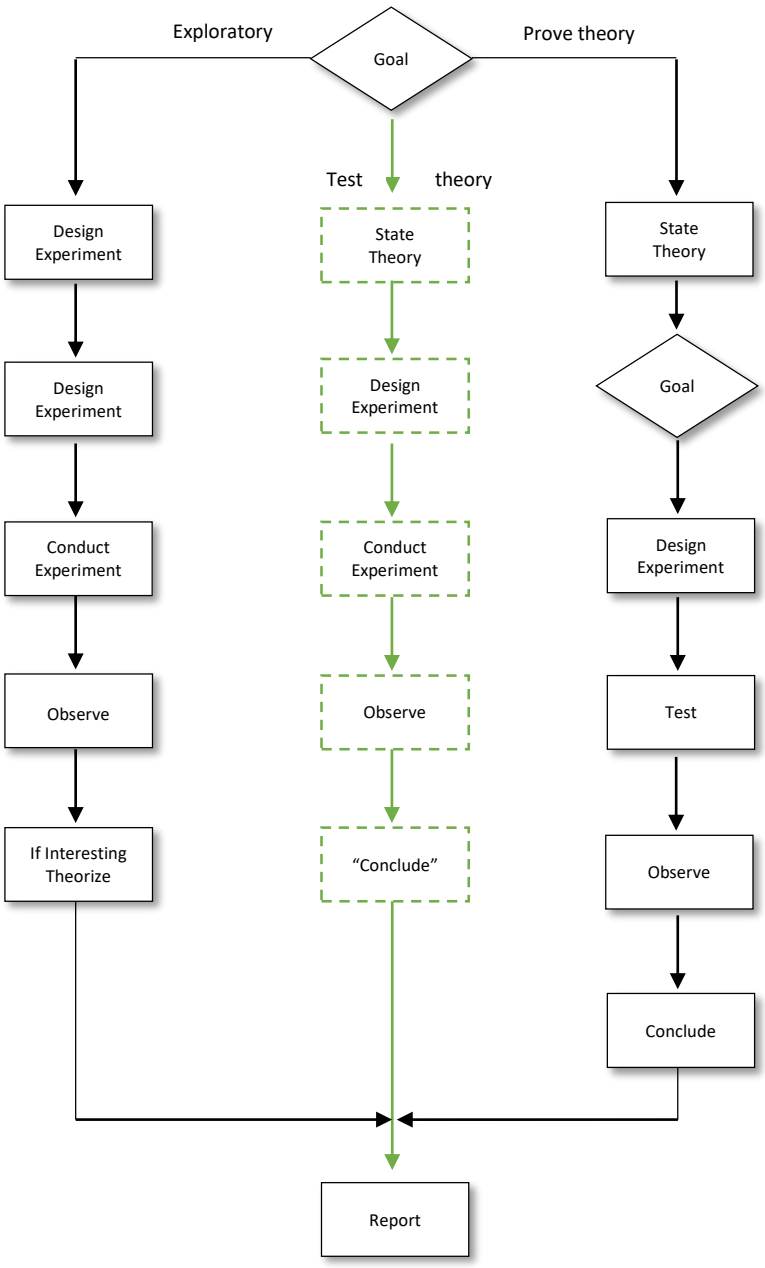
The next task in the Planning and Design phase is the experimental design. When designing the experiment, several factors were considered, such as:

- the hypothesis that needed to be tested;
- the research objectives of this study;

- the variables to be controlled and quantified;
- the research environment; and
- the internal and external validity of the research.

Fundamentally, there are three approaches to experimental research: exploratory, to prove a theory, or to test a theory (Olivier, 2009). In terms of the research objectives the 'test a theory' experimental approach was adopted as illustrated in Figure 5.3 (Goal versus experiment approach). This experimental approach conforms to the quantitative strategy of this study by producing meaningful empirical observations of first person generated data. All experiments contain variables, which are categorised as either dependent (experimental variables) or independent variables (McLeod, 2019). Independent variables affect one or more dependent variables and can be modified by the researcher. Dependent variables, however, are measurable and are reliant on the conditions of the independent variables (McLeod, 2019; Oates, 2006).

Subsequently, by implementing the same dataset within the same controlled experimental evironment, the dependent variables were defined and used to assess the impact of the independent variables in terms of accuracy, precision and predictation rates. The design of the prototype, establishing a controlled environment, and the identification of dependent and independent variables, are discussed in greater detail in the following sections.



Source: (Olivier, 2009)
Figure 5.3 – Goal versus experiment approach.

Once the input data has been gathered and pre-processed, and the controlled environment and design of the prototype are established, the research progresses to the Implementation Phase. During this phase the concepts outlined in the previous phases are made operational through the development of a prototype using Spyder (a Scientific Python Development Environment) and using appropriate NLP pre-processing, feature engineering and classification algorithms to generate the experimental data.

The fundamental objective of experimental research is that there is full control over the environment and this allows for the evaluation of the different independent variables to demonstrate actual performance and efficacy within the same environment. Factors such as microprocessor speed, integrated development environment (IDE), programming language and memory do not, therefore, affect the outcome of any of the experiments conducted.

Consequently, the development and application of the same prototype environment used to conduct the series of experiments, certifies that a controlled environment was achieved. Ensuing the execution of each experiment, all empirical evidence generated was appropriately recorded.

After conducting the experiments, the research progresses to the final phase of the model i.e. the Evaluation phase where the results, evaluation and deductions are presented and discussed in Chapter 6. The final step of the Evaluation phase is deliberated in Chapter 7 of this study which comprises the contributions and future research related to the field.

5.4 Research Methodology

The previous section provided an overview of the research design model, with further discourse on the experimental approach (i.e. 'to test a theory'). The following subsections present a detailed discussion on the implementation of the research design model including data gathering and preprocessing steps, the nature of research environment, variable identification and the effective operation of concepts through the use of a constructed prototype.

5.4.1 Data gathering

In this subsection the data-gathering steps employed by the researcher to build the appropriate corpora for this study are discussed. Given the rationale and hypothesis of this study it is vital that cognizance is taken of the nature and composition of the input dataset demanded by the elected research method. To achieve this, Objective 2 (OB2) of the study is referred to: **test the ability of an NLP-based prototype through a series of controlled experiments, to discover its ability to detect stegospam generated using a probabilistic context-free grammar (PCFG) or linguistic steganography.**

As discussed in Chapter 3 of this study, context-free probabilistic grammars are commonly used in summarisation tasks and for the generation of context-free text corpora, through the use of random and statistical generation methods (Koluguri, Gouse and Reddy, 2014). Implementing NLP algorithms to perform computational tasks for the analysis and classification of stegospam generated, using one or more levels of linguistic steganography would require that the input dataset be in plaintext format with additional preprocessing needed before parsing it through the constructed prototype.

Now that the rationale for the format of the input dataset has been established, we need to identify the size and source of the dataset. For this study, the distinction between the input dataset and the corpus needs to be defined. Therefore, reference to 'corpus' would indicate that the data gathered is still raw and has not been preprocessed in anyway whatsoever, and reference to 'input dataset' or 'dataset', would indicate that the corpus has undergone some form of preprocessing to ensure that it complies with the format required by the NLP-based constructed prototype.

For the corpus the following structure was used, this was to ensure that the experiments were conducted using an adequate sample size of non-stegospam (traditional spam files that do not contain any hidden messages) and stegospam (spam files generated using linguistic steganography).

Type of Spam	Size	Tool or Source	File Format	Secret length
Non-stegospam	800	100 – spam emails from a personal email account 700 – spam emails from SpamAssassin (https://spamassassin.apache.org/old/publiccorpus/)	Plaintext (.txt)	5 – 20 words
Stegospam	800	Self-generated stegospam using random secret messages via the PCFG - Spammimic (https://www.spammimic.com/)	Plaintext (.txt)	None

Table 5.1 – Composition of the Corpus

The composition of the non-stegospam corpus was designed to ensure that it had a decent assortment of spam emails. SpamAssassin is a publicly available email corpus hosted by Apache.org, which is suited for testing and research purposes (Mason, 2006). The composition of the stegospam corpus was generated using the online PCFG tool –

Spammimic. This was achieved through the use of a simple Python script developed using the Selenium library (a Python-based programming library designed for web associated tasks) to automatically populate Spammimic with random secret messages of varied lengths (between 5 - 20 words) in order to generate the stegospam corpus. Whilst other linguistic steganographic tools or algorithms were available, Spammimic was elected due its popularity, storage capacity and imperceptibility.

5.4.2 Ethical considerations of the corpora

The stegospam corpus was generated using random phrases of varied length to symbolise secret messages in order to generate the stegospam files using the PCFG - Spammimic. The tedious process of entering 800 secret messages into Spammimic to generate 800 stegospam files was automated using a Python script. Additionally, the stegospam files generated were used solely for the purpose of this study and no further analysis or distribution of the files was conducted.

The non-stegospam corpus was composed of 100 spam emails received via the researcher's personal email account and 700 from the publicly available SpamAssassin corpus. The SpamAssassin corpus is a selection of email messages, suitable for use in testing spam filtering systems and for research purposes (Mason, 2006). All spam email headers were reproduced in full and some address obfuscation was made with hostnames in some cases having been replaced with '*spamassassin.taint.org'* (Mason, 2006). Further to obfuscation, all metadata from both stegospam and non-stegospam files were scrubbed using the Python library Scrapy. No further analysis or usage of the corpora outside the jurisdiction of this study was conducted.

5.4.3 Data pre-processing

In any Machine Learning (ML) process including Natural Language Processing, data pre-processing is that step in which the unstructured raw corpora are transformed into a state that allows the independent variables of the proposed steganalysis to easily parse it. In other words, after pre-processing, the features of the data can be easily mined and analysed through the implementation of an appropriate algorithm (Pranjal, 2019). Pre-processing

includes several techniques such as scrubbing, integration, transformation, and reduction (discussed in Chapter 4, Section 4.3.1) to produce the required input dataset (Alasadi, 2017).

A dataset is regarded as a collection of data objects, which are often also referred to as records, vectors, patterns, events, samples, observations, or entities. Data objects are defined by a number of features (such as variables, characteristics, fields, attributes, or relationships) that capture the basic property or characteristics of an object, such as the mass of a physical object or the time at which an event occurred, among others (Pranjal, 2019).

Pre-processing the stegospam and non-stegospam corpora comprise two distinct processes:

- Pre-processing the corpora to ensure that only the plaintext contents from both the stegospam and non-stegospam files were extracted for the construction of the input dataset. This was achieved by implementing the Python library – Scrapy, to scrub all metadata from both spam file types.
- After scrubbing off the metadata, the plaintext contents of each spam file type was stored in a single comma-separated (.csv) file named 'data.csv', using the structure {'text', label}. This is shown in Table 5.2 (Structure of the data.csv file) below. This structure is an important requirement for the constructed prototype. Once populated with both structured corpora, the data.csv file is now referred to as the input dataset.

Spam type	Content Format	Label	Description
Non-stegospam	Plaintext or String	0	The label" 0" indicates that the spam file is a non-stegospam file.
Stegospam	Plaintext or String	1	The label" 1" indicates that the spam file is a stegospam file.

Table 5.2 – Structure of the Data.csv file

Now that the first step is complete the input dataset is ready for the next data pre-processing step.

- As with all machine learning tasks and in particular Natural Language Processing, the dataset must undergo additional pre-processing to allow it to be easily parsed into the independent variables of the proposed steganalysis for further processing such as

feature engineering and classification. This is achieved by importing the dataset from the data.csv file into the constructed prototype and subjecting it to the following NLP-based preprocessing tasks (these were discussed in Chapter 4, Section 4.3.1):

- Removal of unnecessary whitespace, stopwords, punctuations, URLs and hashtags.
- Sentence Tokenization using NLTK (Natural Language Toolkit) and
- Lemmatization of words using NLTK.

Upon completion of the second pre-processing step the dataset processed is now ready for further processing and classification.

5.4.4 Experimental Environment

The preceding subsection discussed the pre-processing tasks required for preparing the input dataset for further processing and classification by the constructed prototype. To test the hypothesis, all experiments must be conducted within a controlled environment that employs the same parameters and dataset in order to generate valid empirical evidence. Implementing a controlled experimental environment helps maintain the integrity of the cause-and-effect relationship associated with the positivistic paradigm (Marczyk, DeMatteo and Festinger, 2005). Consequently, all experiments were conducted using a single stand-alone computer with all the required programing classes and libraries pre-installed. The hardware and operating system specifications of the system used to host the experimental environment is detailed in Table 5.3 (System specifications of the Experimental Environment) below.

Component	Specification
System Model	HP EliteBook Folio 9470m
CPU	Intel® Core™ i7-3667U CPU @ 2.00Ghz 2.50GHz
Random Access Memory (RAM)	12.0 GB – Kingston 1333Mhz
Hard Disk	Toshiba - 1TB SATA - 5400 RPM
Motherboard	Intel Motherboard
Chipset	Intel QM77
Operating System	Microsoft Windows 10 Enterprise (64-bit)

Table 5.3 – System specifications of the Experimental Environment.

The development tools used to gather the corpora and construct the steganalysis prototype are detailed in Table 5.4 (Development Environment and specifications) below.

Component	Specification
Integrate Development Environment	Spyder 4.0.1 - Scientific Python Development Environment
ASCII text Editor	Notepad ++ v7 (64bit)
Python Version	Python 3.7.6
Core Python Libraries	Selenium, NLTK and Sklearn

Table 5.4 – Development Environment and specifications

Once the hardware and software that make up the experimental environment are established, the design and anatomy of the prototype and the role it plays in this study need to be outlined. This is discussed further in the following subsection.

5.4.5 Anatomy of the Prototype

Any research modelled on the experimental methodology is typically characterised by:

- observation and measurement;
- proving or disproving a relationship between two or more factors;
- identification of the causal factors, prediction, and repetition.

Hence, the implementation of a prototype is an accepted approach used by researchers to conduct experiments within the field of IS (Oates, 2006). As mentioned in the previous subsections, the construction of the prototype is purely used to test the hypothesis which declares that the implementation of NLP techniques can detect or classify stegospam generated using linguistic steganography.

Consequently, the rationale for the development of the steganalysis prototype is to employ NLP methods to test the hypothesis. Whilst there are just a handful of linguistic steganalysis tools available, they lack the design and underlying NLP techniques required by this study to test the hypothesis. These linguistic steganalysis approaches are discussed in greater detail in section 5.8 (Related Research).

Detecting linguistic steganography is an extremely difficult task, which is attributed to the high level of imperceptibility of the concealed message and the data transformation

techniques used to generate the stego-text (Almohammad, 2010). A steganalysis prototype constructed on various natural language processing algorithms and methods requires that the input dataset be preprocessed in a manner that allows it to be easily parsed for further processing including the mining of deep and meaningful features that can be used to accurately classify stegospam from non-stegospam files.

There were several factors that were considered for the construction of the prototype, these included:

- **Nature and composition of the input dataset.** The input dataset for this study contains stegospam files that were generated using a PCFG and hence does not contain the original text to conduct any form of statistical and comparative analysis, as several steganalysis studies have done, to discover semantic, statistical or syntactic deviations in the stego file generated.
- **Feature Engineering Module.** The feature engineering algorithms or approaches employed by similar studies on stego-text, were fundamentally observational (discussed in section 5.8) and hence lacked deeper word embeddings, meaningful relationships and mathematical similarities between the words and sentences of the stego-text generated. Consequently, the features mined were too specific to the originating steganographic tool or stego-text itself and could not be generalized.
- **Classification Module.** The classification module implemented by the prototype must be able to generate a high degree of accuracy when comparing stegospam against non-stegospam files using the features mined from the Feature Engineering module.
- **Data Analysis and Quantification.** The prototype must be able to perform quantitative and statistical analysis on the experimental data generated to allow for tangible deductions to be drawn.

Given the above considerations, the framework illustrated in Figure 5.4 below was used for the construction of the prototype. The framework permits for the substitution of feature engineering algorithms, classification modules and preprocessing techniques to generate comparative experimental data. This presents the opportunity to test the hypothesis using different algorithms to discover the most effective and accurate approach through causality.

Causality denotes that one event will result in the occurrence of another, subsequent event. In other words, it relates to cause and effect (Decarlo, 2018).

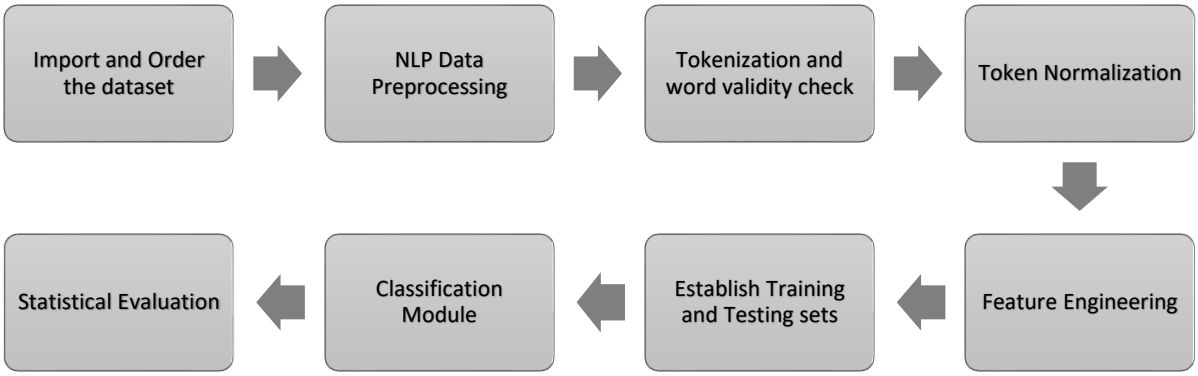


Figure 5.4 - Framework for the Steganalysis process

The following paragraphs provide a narrative of the various steps taken in the steganalysis process as defined in the above framework and its associated programming abstract.

During the first step of the steganalysis framework the input dataset from the 'data.csv' file was imported into the prototype using Pandas (a Python-based software library used for data analysis and manipulation (McKinney, 2020). As previously discussed in subsection 5.4.3, the dataset is in the format {'text', label} and illustrated in Table 5.5 below, image ID#1.

Prototype Framework Step	Pseudo-Code / Programming Abstract	Image ID	Corresponding Prototype Output/Image
Importing and Ordering the Dataset	`None`	**ID#1**	C:\Users\FHChohan\Documents\June 2020 Backup\A_UNISA 2020\Final Code\MyCodes\data.csv - Notepad++ File Edit Search View Encoding Language Settings Macro Run Plugins Window ? data.csv new 1 1 Text,Label 2 "Dear Friend ; Thank-you for your interest in our publication 3 . If you no longer wish to receive our publications 4 simply reply with a Subject: of ""REMOVE"" and you will 5 immediately be removed from our club ! This mail is 6 being sent in compliance with Senate bill 1627 , Title 7 5 , Section 306 ! This is NOT unsolicited bulk mail 8 ! Why work for somebody else when you can become rich 9 within 10 months . Have you ever noticed nobody is 10 getting any younger plus people love convenience ! 11 Well, now is your chance to capitalize on this . We 12 will help you decrease perceived waiting time by 200% 13 & sell more . You can begin at absolutely no cost to 14 you . But don't believe us ! Mrs Jones who resides 15 in Massachusetts tried us and says ""Now I'm rich many

	#Importing the Dataset Spam_data = read ('data.csv') #Calculate and print the number of valid data points {...} #Print Labels and split {...} #Calculate the number of stegospam files and spam file{...}	**ID#2**	[nltk_data] Downloading package wordnet to [nltk_data] C:\Users\FHChohan\AppData\Roaming\nltk_data... [nltk_data] Package wordnet is already up-to-date! There are 1600 data points. 0 Text 1 Label 1 0.5 0 0.5 Name: Label, dtype: float64 1 800 0 800
	#Ordering the dataset #Create Pandas DataTable {...} #Import text and labels into separate fields{...} #Create length field {...} #Calculate the length of each file and store in the "length" field{...}	**ID#3**	Text Label length 0 Dear Friend ; Thank-you for your interest in o... 1 4839 1 Unbelievable Prices On Cell Phones And Accesso... 0 842 2 Dear Friend ; Your email address has been subm... 1 2844 3 My name is Wayne Harrison and I would like to ... 0 1415 4 Dear E-Commerce professional ; This letter was... 1 4620

Table 5.5 – Steganalysis framework -Importing and Organising the dataset

The dataset contained a total of 1 600 spam files of which 50% were stegospam files and 50% non-stegospam files. This is illustrated in Table 5.5, Image ID#2 above, where it can be seen that the dataset has been imported into the prototype and has identified 1 600 data points i.e. 1,600 spam files. This was achieved using the corresponding pseudocode. Image ID#2 further reveals that the data points were equally divided into two categories labelled 1 and 0, where 1 represents the stegospam and 0 representing the non-stegospam files. The categorisation of the spam files was established using predefined labels assigned to each file in the input dataset. Assigning predefined labels is an important step in supervised classification of data as discussed in Chapter 4, section 4.5. The prototype then proceeds to organise the dataset into a Pandas dataframe, calculates the length of each file and stores it into a field titled 'length', shown by the pseudocode and illustrated in Figure 5.5, Image ID#3. This affirms the experimental strategy to include files of varied lengths within the dataset.

Once the input dataset is imported and organised into a data frame the prototype progressed to the next step, which according to the framework, is the NLP data preprocessing step. A detailed discussion on the second phase of data preprocessing was presented in Section 5.4.3

of this chapter. The NLP preprocessing is essential to ensuring that the data being parsed into the feature engineering module for feature mining is clean of any 'noise' such as URLs, punctuation, stopwords, special characters and so on. This step was handled by the `ProcessText(data)` method as shown in the pseudocode for which the corresponding output is illustrated in Table 5.6, Image ID#1 shown below. Here it can be seen that the sentences have been scrubbed of punctuation marks, special characters, URLs and so on using the NLTK software library.

Prototype Framework	Pseudo-Code / Programming Abstract	Image ID	Corresponding Prototype Output/Image

NLP Data Preprocessing	`#Pre-processing the Corpus using NLTK`
	`def processText(data):`
	`  dataset_ch = Lower ()`
	`  #Remove urls {...}`
	`  #Remove usernames {...}`
	`  #Remove whitespace {...}`
	`  #Remove hashtags{...}`
	`  #Remove stopwords{...}`
	`  #Remove punctuation {...}`
	`  #remove \n {...}`
	`return data`
		ID#1	`1595  gooday\r\rwith warm heart friend send greeting...`
			`1596  hi \r\ri currently netherlands seeking politic...`
			`1597  find prospects bizproductsfast \r\rdo know use...`
			`1598  good day sirmadam\r\rplease find attached broc...`
			`Name: Text, Length: 1600, dtype: object`
			`0  [dear friend  thankyou interest publication  l...`
			`1  [unbelievable prices cell phones accessoriesha...`
			`2  [dear friend  email address submitted to us in...`
			`3  [name wayne harrison would like share agenuine...`
			`4  [dear ecommerce professional  letter specially...`

Table 5.6 – Steganalysis Framework - Data preprocessing using NLP Techniques

Succeeding data scrubbing, the dataset was then tokenized into sentences and then further into discrete words or tokens. Sentence and Word tokenization is a subtask of the morphological phase of any NLP pipeline and is a well-established process of Artificial Intelligence (AI), as discussed in Chapter 4, Section 4.3.1 (Bird, Klein and Loper, 2009). Tokens are typically individual words that are used as input for various NLP tasks. Tokenization represents a form of pre-processing and the identification of basic units or tokens to be processed (Geitgey, 2018). The pseudocode for the tokenization process and the corresponding prototype output for this step is illustrated in Table 5.7 below.

Prototype Framework	Pseudo-Code / Programming Abstract	Image ID	Corresponding Prototype Output/Image

Data Tokenization	`#transfer text into a set{...}`
	
	`Def Sent_Tokenize(data)`
	
	`#Use NLTK to Tokenize each file into sentences {...}`
	
	`#Use NLTK to tokenize each sentence into words/tokens {...}`
	
	`#Verify each token generated against WordNet`
	
	`return data`
		ID#1	`Name: Text_Sentences, Length: 1600, dtype: object`
			`0 [dear, friend, interest, publication, longer, ...`
			`1 [unbelievable, price, cell, phone, free, ear, ...`
			`2 [dear, friend, email, address, submitted, u, i...`
			`3 [name, wayne, harrison, like, share, risk, opp...`
			`4 [dear, professional, letter, specially, select...`

Table 5.7 – Steganalysis Framework - Data Tokenization

The next step in the steganalysis framework is token normalization, for which the tokens generated were lemmatized. As mentioned in Chapter 4, Section 4.3.1, there are two approaches to token normalization commonly associated with natural language processing - lemmatization and stemming (Bird, Klein and Loper, 2009). The purpose of lemmatization is to reduce the inflectional and derivationally related forms of a word to its common base form. Lemmatization uses a vocabulary and morphological analysis to return the base or vocabulary form of a word, termed a' lemma'. Stemming however, uses a heuristic approach that simply replaces the suffixes of words to obtain its root form, termed a stem (Jabeen, 2018).

For the constructed prototype, lemmatization was preferred as opposed to stemming due to the fact that lemmatization returns a complete and meaningful word that can been verified against a vocabulary (Bird, Klein and Loper, 2009). There were several Python software libraries considered to handle lemmatization which included TextBlob, Gensim Lemmatizer, CliPS and SpaCy. However, the NLTK Lemmatizer was the option taken as the prototype was fundamentally constructed on the NLTK library. Further, NLTK uses the WordNet vocabulary which was used for token verification. WordNet is a large, freely and publicly available lexical database for the English Language aiming to establish structured semantic relationships between words (Prabhakaren, 2018). Table 5.8 below presents the pseudocode and the corresponding prototype output associated with the lemmatization of the dataset.

Prototype Framework	Pseudo-Code / Programming Abstract	Image ID	Corresponding Prototype Output/Images

Token Normalization	`#Lemmatization of the words`
	`#load the NLTK lemmatizer`
	
	`def lemm_list(email_data)`
	`#import tokens {...}`
	
	`#lemmatizer.lemmatize(tokens)`
	`{...}`
	
	`Return (lemmatized_List)`
	
	`#print (lemmatized_List){...}`
		ID#1	`1595 [warm, heart, friend, send, greeting, hope, le...`
			`1596 [hi, i, currently, netherlands, seeking, polit...`
			`1597 [find, prospect, do, know, use, email, course,...`
			`1598 [find, prospect, do, know, use, email, course,...`
			`1599 [good, day, please, find, attached, brochure, ...`
			`Name: lemmed_words, Length: 1600, dtype: object`

Table 5.8 – Steganalysis Framework – Token Normalization

After the tokens of the dataset were lemmatized to their root form, the dataset was primed for the next step of the steganalysis framework, which is feature engineering. Up until this point, the dataset has been simply prepared to ensure that it was in a state suitable for feature extraction.

Before reviewing the pseudocode and prototype output of the feature engineering step, it is vital that the definition of a feature within the context of natural language processing is established. A feature is defined as something that might be considered meaningful in our text corpus, which can be used to classify it (Tazzyman, 2020). Consequently, defining the right features and deciding how to encode them for a learning algorithm can have an enormous impact on the algorithm's ability to extract a tangible feature set. A feature set is a dictionary of all the features extracted by the mining module, which can then be used to train a classifier (Bird, Klein and Loper, 2009). Feature extraction is discussed in greater detail in Chapter 4, Section 4.4.

Similar studies related to the detection of stego-text, were based on feature sets that were fundamentally observational and too specific to the stego-text or steganographic tool itself. Consequently, the feature sets constructed were not able be to classify stego-text generated using other steganographic tools. Some of the key qualities of the positivistic paradigm is objectivity, repeatability and reliability in the empirical evidence produced. Therefore, features mined fundamentally on the observations of the researcher lack deeper statistical or mathematical correlations, which may yield distortions and bias in the experimental data generated and hence, is not suited to a research study that is underpinned by the positivistic

paradigm (Oates, 2006). Consequently, the approach adopted by this study was to implement two different feature engineering algorithms (considered as one of the independent variable for this research) using two instances of the same prototype to mine deeper and meaningful statistical and mathematical correlations between the words and sentences of the corpora. The two feature engineering modules adopted include Word2Vec and Doc2Vec.

These feature extraction models were discussed in greater detail in Chapter 4, Section 4.4, however, a brief overview is presented here:

- **Word2Vec** - is a two-layer neural net that processes text by *'vectorising'* words. The input for Word2Vec is a text corpus and its output is a vocabulary of vectors or neural word embeddings. Neural word embeddings are words with assigned numerical values. Whilst Word2Vec is not a deep neural network, it turns text into a numerical form that deep neural networks can comprehend. Once the vectors have been established, Word2Vec groups vectors of similar words together in what is known as a vector space. This is achieved through the detection of mathematical similarities between the vectors. Word2Vec in itself is an unsupervised learning module and does not generate any features influenced by the researcher therefore preserving its internal integrity (Nicholson, 2019).
- **Doc2Vec** – is an unsupervised NLP learning module that is based on the paragraph vector algorithm, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents (Mikolov & Le, 2014). Doc2Vec works similarly to Word2Vec, however, it assigns numerical values to documents to learn the document's representation as opposed to just words. As with Word2Vec, Doc2Vec groups vectors with mathematical similarities into the same vector space (Ma, 2018).

Prototype Framework	Pseudo-Code / Programming Abstract	Image ID	Corresponding Prototype Output
Feature Engineering	**#Feature Engineering** #Load and configure Feature Mining Module {...} #Establish labels as per Pandas DataFrame #Configure the feature mining parameters #Import preprocessed corpus from previous step #Initialize the model and build feature set/vocabulary		1.816984266042709351e-01 -2.384911663830280304e-02 2.357471548020839691e-02 2.177630215883255005e-01 -3.535498976707458496e-01 -5.484521985054016113e-01 5.626263469457626343e-02 5.076236128807067871e-01 6.044279038906097412e-02 1.321276556700468063e-02 2.868032157421112061e-01 2.764126285910606384e-02

Table 5.9 – Steganalysis Framework – Feature Engineering

Table 5.9 above, presents the pseudocode and a fraction of the corresponding vectors generated using the Doc2Vec feature extraction module from the Gensim[34] library. Similarly, additional experiments were conducted using Word2Vec. The results generated through these experiments are presented and discussed in the following chapter (i.e. Chapter 6 – Experimental Results).

Once mined using the different feature extraction algorithms (i.e. Word2Vec or Doc2Vec), each feature set was used to classify the data points contained in the training and testing tests. Consequently, this corresponds with the next step in the steganalysis framework i.e. to establish the training and testing sets. Defining the training and testing sets is an integral part of the steganalysis process, which ensures that its classifiers are able to learn the differences between the two file types. This then allows it to efficaciously detect or classify a stegospam from a non-stegospam file. Therefore, each experiment conducted used the identical input dataset, initially split into an 80/20 ratio, i.e. 80% for the training set and 20% for the testing set as illustrated in Figure 5.5 (this ratio will be adjusted in order to test the ability of the proposed solution and the causal effect of the independent variables). Assigning a larger training set increases the accuracy and precision of the classifier implemented by the prototype (Bird, Klein and Loper, 2009).

[34] *https://pypi.org/project/gensim/*

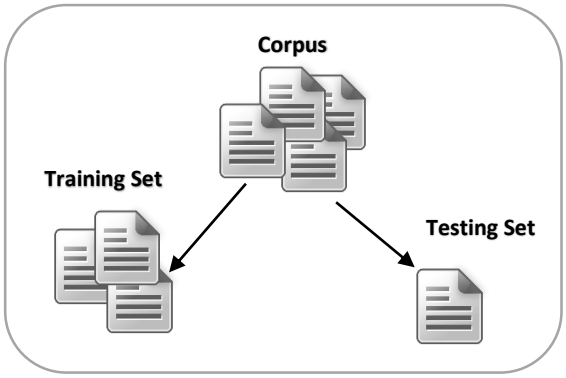


Figure 5.5 – Organisation of the Corpus Data for training supervised classifiers. The corpus is divided into two sets: the training and the testing sets

The next step in the steganalysis framework is the classification (learning) module. As previously mentioned, the objective of this study is to test the hypothesis, which states that the implementation of NLP techniques can detect or classify stegospam files. Therefore, the application of a suitable classifier is key to the ability of the prototype to effectively test the hypothesis and make predictions thereof. Quantitative research is underpinned by the positivistic paradigm which demands repeatability, generalisation and precision in the control and measurement of variables (McGrath, 1982). Hence, to test the precision and elasticity of the prototype, two supervised classifiers were used to evaluate which produced the best results. The classifiers implemented were the Naïve Bayes (NB) and the Support Vector Machine (SVM). These classifiers were discussed in greater detail in Chapter 4, Section 4.5. Additionally, the implementation of more than one classifier is to evaluate the efficacy of each, which supports the research objective stated earlier in the chapter. The corresponding pseudocode for this step is presented in Table 5.10 below; however, the prediction rate achieved by each classifier is presented and discussed in Chapter 6.

Prototype Framework	Pseudo-Code / Programming Abstract	Image ID	Corresponding Prototype Output/Images
Setting the Training and Testing Sets **and** **Classification**	**#Classification Module** #Split Corpus into Training and Testing sets #train the features model Def get_vectors() {...} #Get vectors from trained model #Initial NB and SVM Classifiers Def test_clf() {...} #Execute classifiers #Print prediction rating		Outputs and findings discussed in the following chapter

Table 5.10 – Steganalysis Framework – Classification Module

The final step in the steganalysis framework is Statistical Evaluation. In order to gauge the accuracy and efficacy of the classification models, they must be evaluated to not only satisfy the data analysis requirement associated with quantitative research, but also act as a guide in making improvements to the model (Bird, Klein and Loper, 2009). This step falls within the Evaluation phase of this study and likewise discussed in greater detail in Chapter 6.

5.4.6 Dependent and Independent Variables

A variable is defined as a placeholder and hence can be assigned different values. For example, the speed of a vehicle or the weight of an item can be considered as variables because there are varying speeds and weights. By contrast, if something cannot vary or take on different values, then it is referred to as 'a constant'. When discussing variables, perhaps the most important distinction is between independent and dependent variables.

An independent variable is a factor that is mutable and can be controlled by the researcher (Marczyk, DeMatteo and Festinger, 2005). Independent variables affect one or more dependent variables in relation to a particular property such as its size or length. Whilst a dependent variable is the measure of the effect (if any) of the independent variable (Oates, 2006). In other words, the value of a dependent variable fluctuates as a result to the vicissitudes to the independent variable. In experimental research, an experiment can be

constructed based on the manipulation of the independent variables, in order to observe the effect, it has on the dependent variables (Oates, 2006; Marczyk, DeMatteo and Festinger, 2005). This supports the causal effect where changes to an independent variable induces a direct and isolated effect on its dependent variable(s) (Oates, 2006).

For this study the following independent variables were considered and were modified to affect the results of each experiment: Training and Testing Set Sizes; Feature Extraction Module; and Classification Module.

Independent Variables	Description
Training and Testing Set Size	The size of the training and testing set can be modified, to test its effect on the prediction and accuracy ratings of the NLP-based prototype.
Feature Extraction Module	This variable relates to the supervised feature engineering module used to mine deep statistical and mathematical vectors to generate the feature set. Changing this module can influence the prediction, accuracy, F-Measure and other ratings.
Classification Module	The classification module is used to detect or classify stego from non-stegospam files and can be mutated to test the causal effect on the experimental prediction and data.

Table 5.11 - Prototype - Independent Variables

For this study the following dependent variables were considered. The values of these variables fluctuate due to the causal effect induced via modifications to the independent variables tabulated in Table 5.11 - Prototype - Independent Variables.

Dependent Variables	Description
Prediction Rating	Refers to the probability rate of a classifier for predicting the labels of a test set based on the features mined from the training set.
Accuracy	The simplest metric that can used to evaluate a classifier, accuracy measures the percentage of inputs in the test set that the classifier correctly labelled (Bird, Klein and Loper, 2009).
Precision	Precision is the ability of a classifier not to label an instance as positive that is actually negative. For each class it is defined as the ratio of true positives to the sum of true and false positives. In other words, for all instances classified positive, what percent was correct? (McIntyre, et al., 2019).
Recall	Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. In other words, for all instances that were actually positive, what percent was classified correctly? (McIntyre, Bengfort and Gray, 2019).
F-Measure	The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 are used to compare classifier models, not global accuracy (McIntyre, Bengfort and Gray, 2019).

Table 5.12 - Prototype - Dependent Variables

After modifying the independent variables, several controlled experiments using the same input dataset and experimental environment were conducted. Following which, the results of each experiment were recorded and stored for further analysis and to establish any findings. These results and findings are presented and discussed in Chapters 6 and 7 respectively.

This subsection presented and discussed the key independent variables and the impact they had on the dependent variables which were used to assess the causal effect of each experiment.

5.5 Controls

The primary strength of experimental research is precision and control; its principal purpose is to test and extend theory**.** To achieve this purpose, experiments are designed on the causal effect, wherein changes in the independent variables affect the values of the dependent variables (Oates, 2006).The important implication is that in designing an experiment, one should focus on precision and control above everything else, because precision is the *raison d'être* of experimental research (Dennis and Valacich, 2001).

In this research study, all experiments were conducted in a well-structured and isolated environment where control over the independent variables and the experimental environment itself was preserved. This was achieved by implementing the same input dataset and NLP preprocessed dataframe across all experiments. Further, the same prototype was used across all experiments, which consisted of the same software libraries and natural language processing methods.

As stated, the aim of a research strategy grounded on experimentation is to demonstrate the causal effect, where the independent variables induce some form of effect on the dependent variables (Oates, 2006; Olivier, 2009). To preserve control, the same dependent variables were observed and results recorded.

Experimental research can be conducted in a laboratory or in the field. Laboratory based experiments are considered to have greater control as they are conducted in a well-constructed experimental environment. On the contrary, field studies are conducted in the natural environment where several external factors can influence the research (Oates, 2006). This research study was conducted in a controlled and isolated environment using the same operating system, programming language, software libraries and hardware.

5.6 Reliability and Validity

Denise and Polit (2004), refers to reliability as the degree of consistency with which an instrument measures the attribute it is designed to measure. For the experimental data produced to be considered valid, the measurement criteria or instrument *(prototype)* must first be reliable and aligned to the research methodology. Seeing that the philosophical

paradigm for this research is a positivistic one, reliability in positivistic research is focused on repeatability (Denise and Polit, 2004). Reliability of the prototype was qualified through the implementation of the well-constructed and controlled experimental environment that allowed for the elicitation of repeatable empirical evidence. The same prototype was used across all experiments with only the independent variables modified. The data generated was further subjected to rigorous quantitative and statistical evaluation to ensure the reliability of the prototype. The results and findings of these evaluations are presented in the following chapters.

While reliability is concerned with the accuracy of the actual measuring instrument, validity is concerned with the study's achievement at evaluating what the researcher intended to accomplish. It is important to acknowledge that reliability and validity are not independent, an instrument that is unreliable and inconsistent cannot possibly be valid (Denise F. Polit, 2004). There were two types of validity that were considered for this study these included internal and external validity.

Internal validity refers to the rigor with which the research was conducted (e.g., the research design, the care taken to conduct measurements, and decisions concerning what was and was not measured). According to Oates (2006), an experiment is considered to have good internal validity if the measurements acquired are conclusively attributed to the manipulation of the independent variable, and not to any other factors. Therefore, internal validity can be appraised through a comparative analysis of the results produced by each experiment (Polit, 2004).

Consequently, this research was conducted through a series of controlled experiments using the same hardware and software, with each experiment representing a modification of the independent variables. The results thereof were recorded and a comparative analysis was conducted. This evaluated the cause and effect relationship between the dependent and the independent variables. By doing so, the internal validity of the prototype was assessed.

Marczyk, DeMatteo and Festinger (2005), describe external validity as the degree to which findings are generalisable to different people, settings and times. An experiment is considered

to have good external validity if the results generated are not unique to a particular set of circumstances but can be generalised. An ideal approach to demonstrate generalisability is to repeat the experiment several times in different situations (Oates, 2006).

This research study was conducted using the predefined framework as illustrated in Figure 5.4 (Framework for the Steganalysis Process). This framework supports the generalisation of several factors and independent variables. Most research studies related to linguistic steganography require the original cover text to detect any statistical or syntactical deviations in the stego-object produced, hence limiting its generalisability. This research, however, employed different unsupervised NLP-based learning algorithms which are capable of inferring deep word embeddings through statistical and mathematical multi-level vectorization of any plaintext corpus without any human intervention (Nicholson, 2019) . This allows for a high level of external validity as the research design can be implemented to detect stego spam files generated by any other PCFGs other than the nominated Spammimic (Nicholson, 2019).

This section has discussed several processes associated with the research methodology including data gathering, data preprocessing and the experimental setup. The results gathered were used to evaluate and address the secondary research question 4:

Which Natural Language Processing methods or algorithms are most effective in detecting or classifying stego from non-stegospam files?

Additionally, the internal and external validity required for research were presented with a discussion on how they were applied to this research study. The next section discusses the limitation of this research.

5.7 Limitations

The experimental approach adopted by this research study has a few limitations, which are presented as follows. Firstly, the design of the steganalysis process required two phases of data pre-processing; the first entailed data gathering and creation of the .csv file, and the second involved the NLP-based data pre-processing required to prepare the input dataset for

feature mining and classification. One limitation of the corpus was that it contained only the content of each spam email and did not consider any email headers, embedded images or attachments. In fact, all metadata was scrubbed as this study only considered the cover-text contained in the body of each spam file. Considering that some steganographic methods use the email headers to conceal a secret message (Halip, et al., 2012), scrubbing it would have removed the steganographic component and rendered the stegospam as a non-stegospam file. Inclusion of the email header should be a consideration for future research.

Secondly, the input dataset contained stegospam files generated using Spammimic and did not include files created using any other steganographic tool. This was attributed to the lack of linguistic steganographic tools associated with the creation of context-free stegospam files.

Additionally, the prototype used to test the hypothesis was constructed using the Python programming language and the NLTK software library. The same prototype was used across all experiments with different feature extraction and classification modules, producing an average execution time of approximately 8 minutes per experiment. The prototype was not tested using any other NLP software library (such as StanfordNLP or SpaCy*),* operating system or computing hardware, which may impact the execution time and the results of each experiment.

Another limitation of the research is that it was limited to the English vocabulary and only considered words tested against the WordNet lexical database. The motivation for the WordNet lexical database is that it is widely supported by the NLTK software library. Furthermore, WordNet is a large, free and publicly available English lexical database thus making it a suitable tool for computational linguistics and natural language processing. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations (Fellbaum, 2006). The implementation of other lexical databases to include different languages and perhaps a broader synset would impact the execution time and broaden the results of each experiment.

This section discussed the limitations of the research in terms of the research environment used. It also provided arguments as to why certain decisions were taken and the impact of those decisions. The next section presents related research.

5.8 Related Research

This section presents a discussion on other research conducted that is related to the research presented here, which also attempts to detect linguistic steganography. Additionally, a discourse is provided to highlight any similarities and differences thereof.

Din, Samsudin and Lertkrai (2012), present a framework of components for a Natural Language Steganalysis. In their paper these researchers suggest a broad NLP framework which encapsulates several text and linguistic analysis techniques into a single steganalysis approach such as text manipulation analysis, statistical analysis, semantic analysis, syntax analysis, and lexical analysis. However, they do not provide any detail on the implementation of any these techniques. Additionally, the framework suggests a steganalysis that would be able to detect both formatting and lexical inflictions within a text document for the classification of a stego-text. The research study presented here does not focus on the detection or classification of stego-text based on formatting irregularities, rather this study focuses only on linguistical irregularities. Furthermore, this study implements machine learning algorithms such as Word2Vec and Doc2Vec to mine deep and meaningful feature sets, which are utilized for the classification or detection of stegospam generated using a PCFG (see Section 4.4 of Chapter 4 for more detailed information on Word2Vec and Doc2Vec). Hence, the research methodology adopted by this study has a higher degree of rigor and generalisation when compared to the approach proposed by Din, Samsudin and Lertkrai, (2012).

Xiang, et al. (2018), have presented a synonym-substitution-based steganalysis approach which uses the statistical value of synonyms for the detection of stego-text. Their approach identifies the synonyms within a given text, maps them to synonymous words found in corresponding substitution sets, and words within their context window to build their word embeddings through the implementation of a Skip-gram feature extraction model. These word embeddings are then subjected to a 'context-fitness' test based on its semantic

correlation to assess the relevance of a context word to a specific synonym. Following which Xiang, et al., extracted three features namely: Context Fitness Maximum Rate *(*CFMR*),* Context Fitness Maximum Deviation *(*CFMD*),* and Context Fitness Mean *(*CFM*).* Subsequently, these features were feed into a SVM classifier to analyse the statistical differences between the original text file and the stego-text (generated using Tlex) for the detection of synonym-substitution based steganography (Xiang, et al., 2018). This research study however, focuses on the detection of stego-text generated using a PCFG and not just the detection of those generated using a rudimentary synonym-substitution approach. Additionally, this study implements the machine learning algorithms - Word2Vec and Doc2Vec to build deeper and more meaningful feature sets and provides a comparative analysis between them. Finally, this study implements both the Naïve Bayes and SVM classification models for classification and prediction of stego-text as opposed to the only the SVM classifier implemented by Xiang, et al. (2018)

Taskiran, et al. (2006), proposed a steganalysis approach to detect steganographic sentences generated using the T-Lex steganogaphic tool. T-Lex was first proposed in 1998 by Keith Winstein in his paper 'Lexical Steganography through Adaptive Modulation of the Word Choice Hash', which employs a synonym-substitution approach in order to generate stego-text using a limited synonym database (Taskiran, et al., 2006). Taskiran, et al. then implemented the StanfordParser to establish the sentence boundaries of the corpus file and selected only those sentences that contained at least one verb as being 'syntactically typical' as possible, no further pre-processing was done. Taskiran, et al., then trained a trigram model for sentences and an SVM classifier using the Stanford Research Institute Language Modeling toolkit (SRILM). This allowed these researchers to classify stego-sentences from non-stego sentences based on the statistical probabilities of synonyms contained in each sentence (Taskiran, et al., 2006).

As mentioned previously this research study focuses on the classification of stego-text generated using a PCFG and not just those generated using a synonym-substitution approach. Furthermore, this study differs in the fact that it employs more advanced feature extraction algorithms such as Word2Vec and Doc2Vec to generate more meaningful feature sets required to train a classifier. This study does not implement the SRILM library for the

establishment of sentence boundaries and other NLP tasks, instead it implements the NLTK library. This research study further differs from that proposed by Taskiran, el al., as it provides a comparative analysis study of the SVM and the Naïve Bayes classifiers to determine which approach is more efficacious for the classification or detection of stegospam from non-stegospam files.

This section briefly discussed research similar to this study and highlighted distinct differences to each approach. The conclusion to this chapter is presented in the next section.

5.9 Chapter Conclusion

This chapter describes the research method adopted. The chapter starts with an overview of the research process to provide a background of the different research approaches that exist. It then presents a discussion of why the experiental research method was elected and its objectives in relation to this study.

The research design is then presented and discussed, with arguments offered as to why the experimental method was adopted. An overview is presented on the steps followed to conduct the research.

The research methodology is then explained in detail and information provided on how the experimental method is applied including ethical considerations and key data preprocessing tasks. The research environment is presented together with the data collection process and a discussion on the anatomy of the prototype. Furthermore, a framework for the steganalysis process is presented and discussed in detail. This is followed by the reliability, validity, controls and limitations of this research as well as a discussion of the difference between this research and similar research that exists.

The next chapter presents the results of the experiments.

(Refer to the Addendum at the end of this research paper for instructions on how to access a copy of the source codes, corpora and the Sypder development environment).

6

Experimental Results

This chapter presents an analysis of the empirical data recorded from each experiment conducted in this study. Consequently, this chapter addresses the secondary research question SRQ2: ***Which Natural Language Processing methods or algorithms are most effective in detecting of classifying stegospam from non-stegospam files?*** *and research objective OB4:* ***Evaluate the ability of a proposed NLP-based steganalysis by conducting several controlled experiments on existing corpora known to contain stegospam and non-stegospam files.***

The research methodology, design and experiments conducted for this research study were discussed in the previous chapter. The predication rate and the results for each experiment conducted were recorded and are presented in this chapter. This chapter provides further discourse on the findings and analysis of the results presented in this chapter.

The structure of this chapter is as follows:

Section 6.1 - Observations and Measurements

Section 6.2 - Presents and discusses the results of the various dependent variables for each experiment conducted

Section 6.3 - Provides a discussion on the predication rates and the execution time of each experiment conducted

Section 6.4 - Conclusion

6.1 Observations and Measurement

Experimental research includes quantitative research and data, which were used to measure and observe changes in the dependent variables. Quantitative data implies that the evidence gathered is based on numerical values, which is used and analysed by positivistic researchers (Oates, 2006). Considering that this research study is hinged on the positivistic paradigm, all results and evidence are presented numerically.

The rationale for data analysis is to discover patterns in the empirical data. This is made easier to achieve in this research study through the use of visual aids such as tables and graphs. The statistical data for this research study was generated using the constructed prototype, which offer universal means and measures for evaluating key points and to formulate generalised conclusions (Oates, 2006). The interpretation and findings of the data produced are presented in the next chapter. As discussed in Chapter 5, the prototype was implemented as a vehicle for demonstrating the research in practical terms i.e. to conduct exploratory experiments.

For each experiment executed, changes in the dependent variables were recorded based on the changes of the independent variables. The results were then used for comparative analysis to ascertain the effectiveness of the different NLP feature extraction and classification modules. For each experiment the following measurements were recorded:

Dependent variables:

- Experiment Name, e.g. Word2Vec
- Classifier, e.g. Naïve Bayes
- Test and Training dataset size e.g. 0.2% [20%].

Independent variables:

- Accuracy
- Precision
- Recall
- F-Measure
- Predication
- Duration of each experiment.

Observations were made based on the impact of the changes in the training and testing sets, feature extraction module and the ability of the classifier to accurately predict a stegospam from a non-stegospam file.

6.2 Classification Report and Results

In the previous chapter, the causal effect between the independent and dependent variables of this study was discussed. In this section the empirical data gathered from the different experiments are presented and deliberated. Furthermore, the measurements between experiments are compared to determine which NLP feature extraction and classification module produced the best results and under which conditions. This further facilitates the findings of this research which are presented in the next chapter.

This section is broken-down into different subsections, each of which focuses on a different dependent variable and the influencing independent variables in order to demonstrate the causal effect. The variables, measurements and comparisons are presented in the order: Accuracy, Precision, Recall and F-Measure.

Before the results for each experiment are presented, there are a few fundamental factors that we need to consider, these include – True Positive(TP), True Negative (TN), False Positive (FP) and False Negative (FN) (Miyasato, 2020; Bird, Klein and Loper, 2009).

True Positive - The actual value was True, and the model predicted it as True.

True Negative - The actual value was False, and the model predicted it as False.

False Positive - The actual value was False, and the model predicted True. This is also known as a Type I error.

False Negative - The actual value was True, and the model predicted False. This is also known as a Type II error.

6.2.1 Accuracy

The most fundamental metric used to evaluate a classifier is its accuracy - which measures the inputs in the test set that the classifier correctly labelled. For example, a name gender classifier that predicts the correct name 70 times in a test set containing 90 names would have an accuracy of 70/90 = 78%. (Bird, Klein and Loper, 2009). The formula used to calculate the accuracy of a classifier is:

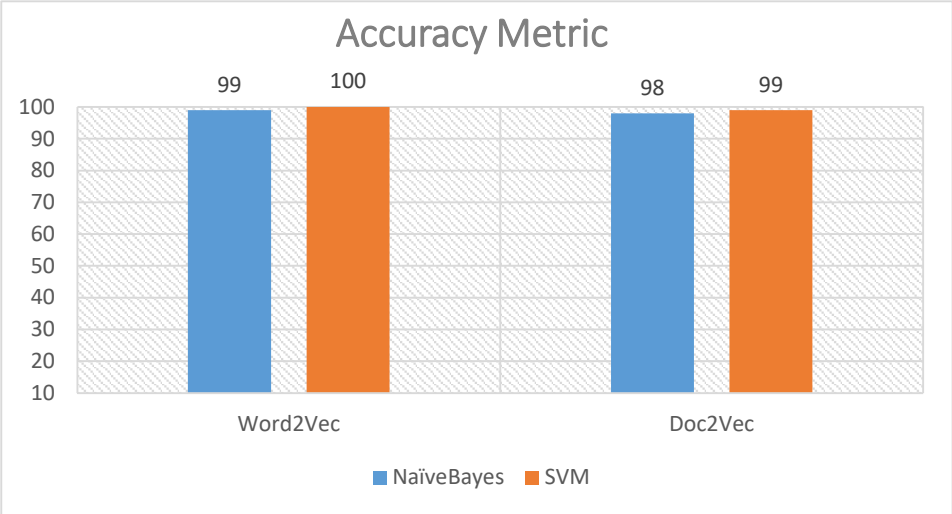
$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Source: (Miyasato, 2020)
Figure 6.1 - Accuracy Formula

The sum of true positives and true negatives divided by the total number of samples. This is only accurate if the model is balanced. It will yield inaccurate results if there is a class imbalance between the training and testing sets (Miyasato, 2020; Bird, Klein and Loper, 2009). The accuracy results for each classifier and feature extraction algorithm observed and documented is shown in Table 6.1 below, generated through a succession of experiments using 20% of the dataset as the testing set and 80% as the training set.

Independent Variables				**Dependent Variable**
Classifier	**Feature Extraction Algorithm**	**Test Set**	**Training Set**	**Accuracy Result**
Gaussian Naïve Bayes	Word2Vec	0.2 [20%] *320 files*	0.8 [80%] *1280 files*	0.99 [99%]
SVM (Support Vector Machine)	Word2Vec	0.2 [20%] *320 files*	0.8 [80%] *1280 files*	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	0.2 [20%] *320 files*	0.8 [80%] *1280 files*	0.98 [98%]
SVM (Support Vector Machine)	Doc2Vec	0.2 [20%] *320 files*	0.8 [80%] *1280 files*	0.99 [99%]

Table 6.1 - Accuracy Metric using an 80/20 ratio for the Training and Testing sets respectively

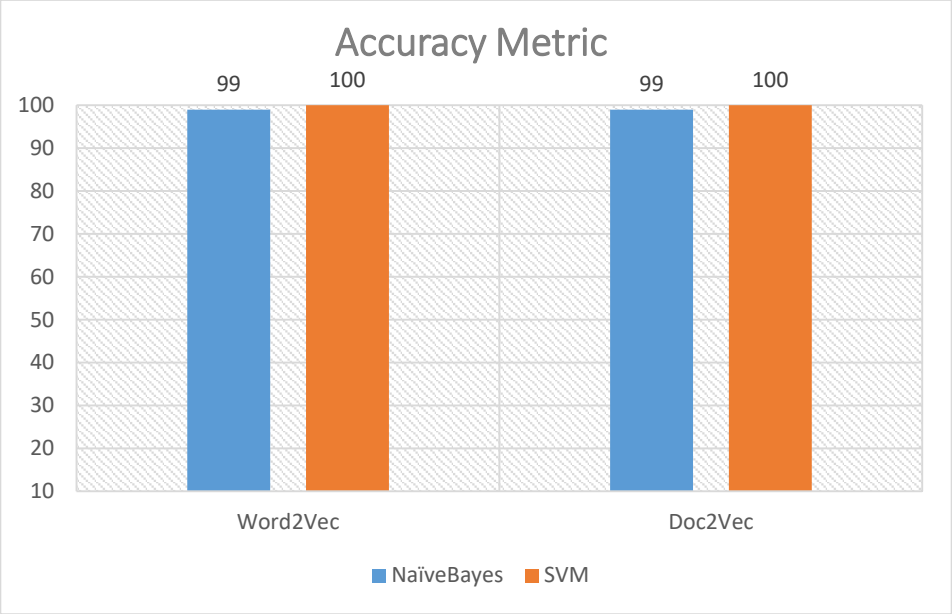


Graph 6.1 – Accuracy Metric using an 80/20 ratio for the Training and Testing sets respectively.

Based on the experimental results generated using 20% of the dataset as the testing set, it was discovered that both the Naïve Bayes and the SVM classifiers achieved better accuracy ratings when implemented using Word2Vec as opposed to the Doc2Vec algorithm. To further evaluate the accuracy metric for both classifiers, the size of the testing set was adjusted to 30% of the input dataset. The results generated and documented thereof are presented in Table 6.2 and Graph 6.2 below.

Independent Variables				**Dependent Variable**
Classifier	**Feature Extraction Algorithm**	**Test Set**	**Training Set**	**Accuracy Result**
Gaussian Naïve Bayes	Word2Vec	0.3 [30%] *480 files*	0.7 [70%] *1120 files*	0.99 [99%]
SVM (Support Vector Machine)	Word2Vec	0.3 [30%] *480 files*	0.7 [70%] *1120 files*	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	0.3 [30%] *480 files*	0.7 [70%] *1120 files*	0.99 [99%]
SVM (Support Vector Machine)	Doc2Vec	0.3 [30%] *480 files*	0.7 [70%] *1280 files*	1.00 [100%]

Table 6.2 - Accuracy Metric using an 70/30 ratio for the Training and Testing sets respectively



Graph 6.2 – Accuracy Metric using an 70/30 ratio for the Training and Testing sets respectively

Based on the experimental results achieved using 30% of the input dataset as the testing set and 70% as the training set we find that both classifiers achieved the same accuracy rating as in the previous experiment when implemented using the Word2Vec algorithm. However, the accuracy ratings for both classifiers improved by 1% each when implemented using the Doc2Vec algorithm. The next dependent variable tested and documented was the precision rating for each classifier implemented using the Word2Vec and the Doc2Vec algorithms.

6.2.2 Precision

Precision is the relation of correctly predicted positive observations to the total predicted positive observations (i.e. true positive + false positive) (Bird, Klein and Loper, 2009). The question that this metric answers is: Of all spam files in the input dataset labeled as stegospam, what proportion of positive predictions was actually correct? In other words, if this study's model has a precision rating of 0.5, whenever it predicts a spam file as stegospam, 50% of the predictions would be correct. The formula to calculate the Precision metric is as follows:

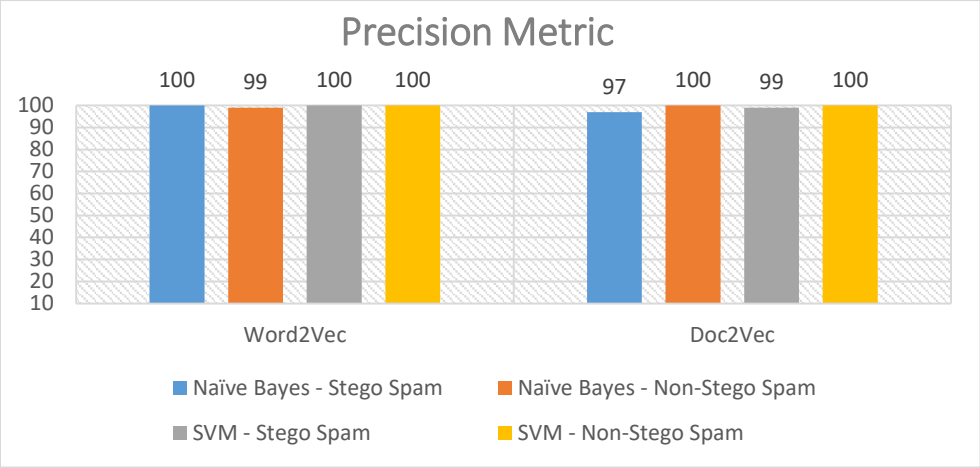
$$Precision = \frac{tp}{tp+fp}$$

Source: (Miyasato, 2020)
Figure 6.2 - Precision Formula

The precision results achieved by each classifier and feature extraction algorithm are presented in Table 6.3 and Graph 6.3, generated via a series of experiments using 20% of the dataset as the testing set and 80% as the training set (i.e. 1 280 files for the training set and 320 files for the testing set).

Independent Variables		Dependent Variable	
Classifier	**Feature Extraction Algorithm**	**Precision Result**	
		Non-stegospam [0]	**Stego Spam [1]**
Gaussian Naïve Bayes	Word2Vec	0.99 [99%]	1.00 [100%]
SVM (Support Vector Machine)	Word2Vec	1.00 [100%]	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	1.00 [100%]	0.97 [97%]
SVM (Support Vector Machine)	Doc2Vec	1.00 [100%]	0.99 [99%]

Table 6.3 - Precision Metric using an 80/20 ratio for the Training and Testing sets respectively.



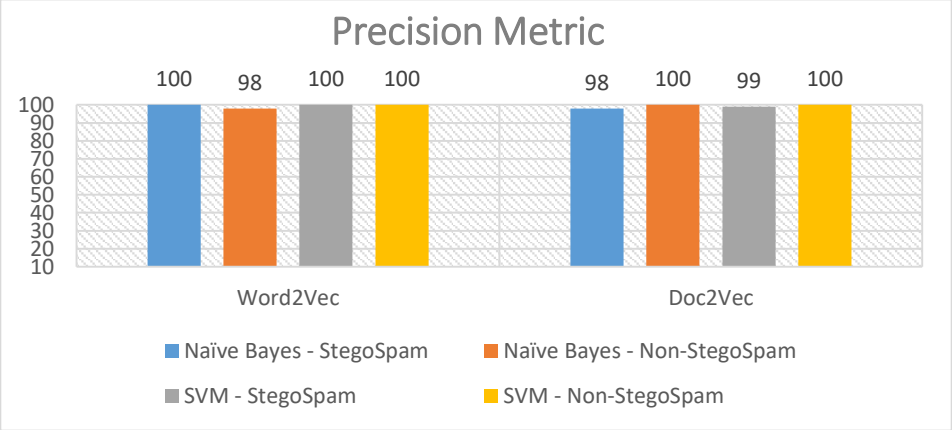
Graph 6.3 – Precision Metric using an 80/20 ratio for the Training and Testing sets respectively

Table 6.3 and Graph 6.3 above demonstrate that the SVM classifier produced marginally better precision ratings than the Naïve Bayes classifier when implemented using either feature extraction algorithms. Additionally, it can be observed that the precision rating for both classifiers is slightly lower when implemented using the Doc2Vec algorithm.

For the next experiment, the training and testing sets were adjusted to a 70/30 ratio (i.e. 1120 files for the training set and 480 files for the testing set). The results achieved and documented from these experiments are presented in Table 6.4 and Graph 6.4 below.

Independent Variables		Dependent Variable	
Classifier	**Feature Extraction Algorithm**	**Precision Result**	
		Non-stegospam [0]	**Stego Spam [1]**
Gaussian Naïve Bayes	Word2Vec	0.98 [98%]	1.00 [100%]
SVM (Support Vector Machine)	Word2Vec	1.00 [100%]	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	1.00 [100%]	0.98 [98%]
SVM (Support Vector Machine)	Doc2Vec	1.00 [100%]	0.99 [99%]

Table 6.4 - Precision Metric using an 70/30 ratio for the Training and Testing sets respectively



Graph 6.4 – Precision Metric using an 70/30 ratio for the Training and Testing sets respectively.

By adjusting the size of the training set to 70% of the input dataset, it was discovered that the precision rating achieved by the SVM classifier remain constant at 100% when implemented using the Word2Vec algorithm. When implemented using the Doc2Vec algorithm the precision rating of the Naïve Bayes classifier marginally improved from 97% to 98% when classifying stego spam files, whilst the SVM classifier precision ratings remained unchanged. These results are discussed further in section 7.2 of Chapter 7. The next dependent variable tested and documented was the recall rating achieved by each classifier when implemented using the Word2Vec and the Doc2Vec feature extraction algorithms.

6.2.3 Recall

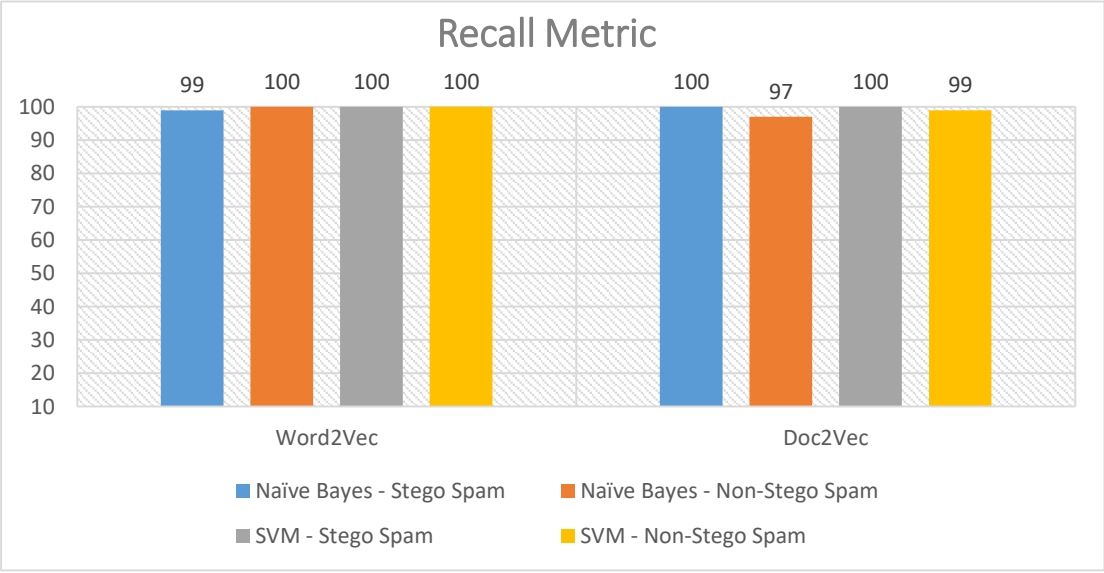
Recall is the ratio of correctly predicted positive observations to all observations in actual class labelled positive. Recall answers the question: Of all the spam files in the input dataset labeled as stegospam, how many was identified correctly? Generally, a rating above 0.5 is regarded as good. A model that has no false positives will have a recall of 1.0 (i.e. 100%) (Miyasato, 2020; Bird, Klein and Loper, 2009). To calculate the recall rating we use the following formula:

$$Recall = \frac{tp}{tp+fn}$$

The recall results achieved by each classifier and feature extraction algorithm are presented in table 6.5 and graph 6.5, generated through a series of experiments using 20% of the dataset as the testing set and 80% as the training set (i.e. 1 280 files for the training set and 320 files for the testing set).

Independent Variables		**Dependent Variable**	
Classifier	**Feature Extraction Algorithm**	**Recall Result**	
		Non-stegospam [0]	**Stego Spam [1]**
Gaussian Naïve Bayes	Word2Vec	1.00 [100%]	0.99 [99%]
SVM (Support Vector Machine)	Word2Vec	1.00 [100%]	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	0.97 [97%]	1.00 [100%]
SVM (Support Vector Machine)	Doc2Vec	0.99 [99%]	1.00 [100%]

Table 6.5 - Recall Metric using an 80/20 ratio for the Training and Testing sets respectively



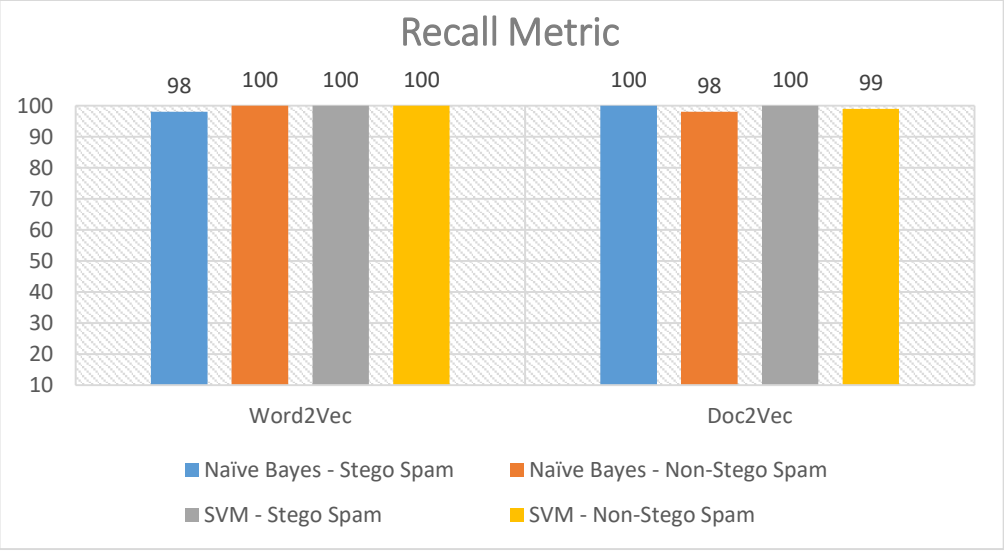
Graph 6.5 – Recall Metric using an 80/20 ratio for the Training and Testing sets respectively

Table 6.5 and Graph 6.5 above demonstrate that the SVM classifier achieved a 1.0 [100%] recall rating when implemented using the Word2Vec algorithm. This infers that when using the Word2Vec algorithm, SVM produces zero false positives. However, its recall rating marginally decreased when implemented using the Doc2Vec algorithm. The Naïve Bayes classifier achieved a slightly lower recall rating when implemented using the Word2Vec and the Doc2Vec algorithms compared to the SVM classifier.

For the next experiment the size of training and testing sets were adjusted to a 70/30 ratio (i.e. 1120 files for the training set and 480 files for the testing set). The results achieved and documented from these experiments are presented in Table 6.6 and Graph 6.6 below.

Independent Variables		**Dependent Variable**	
Classifier	**Feature Extraction Algorithm**	**Recall Result**	
		Non-stegospam [0]	**Stego Spam [1]**
Gaussian Naïve Bayes	Word2Vec	1.00 [100%]	0.98 [98%]
SVM (Support Vector Machine)	Word2Vec	1.00 [100%]	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	0.98 [98%]	1.00 [100%]
SVM (Support Vector Machine)	Doc2Vec	0.99 [99%]	1.00 [100%]

Table 6.6 - Recall Metric using an 70/30 ratio for the Training and Testing sets respectively.



Graph 6.6 – Recall Metric using an 70/30 ratio for the Training and Testing sets respectively.

By adjusting the size of the training set to 70% of the input dataset, it can be seen that the recall ratings for the SVM classifier remain unchanged when compared to the recall ratings it

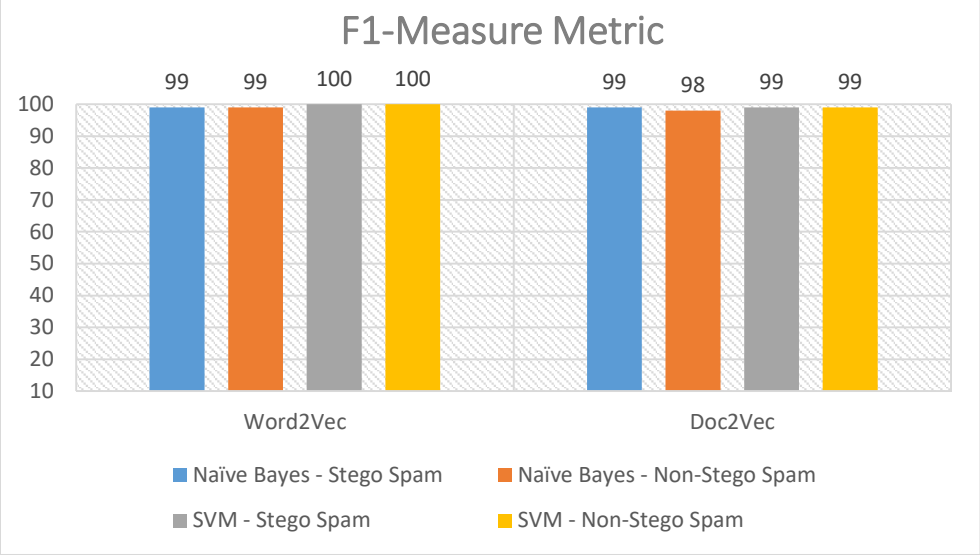
achieved in the previous experiments. The Naïve Bayes classifier, however, achieved fluctuating recall ratings as presented in Table 6.6 and Graph 6.6. These findings are discussed further in Section 7.2 of Chapter 7. The next dependent variable tested and documented was the F1-Measure rating for each classifier implemented using the Word2Vec and the Doc2Vec algorithms.

6.2.4 F1-Measure

The F1-Measure or F1-Score combines the precision and recall to produce a single weighted average, defined as the harmonic mean of the precision and recall metrics. Consequently, this score includes both false positives and false negatives. Intuitively it is not as easy to comprehend as accuracy, but F1 is typically more useful than accuracy, particularly if you have an uneven class distribution. It should be noted that, the weighted average of F1 should be used to compare classifier models, not global accuracy (Bird, et al., 2009; McIntyre, et al., 2019). The F1-measure rating achieved by each classifier through a succession of experiments using 20% of the dataset as the testing set and 80% as the training set (i.e. 1 280 files for the training set and 320 files for the testing set) were recorded as presented below in Table 6.7.

Independent Variables		**Dependent Variable**	
Classifier	**Feature Extraction Algorithm**	**F1-measure Result**	
		Non-stegospam [0]	**Stego Spam [1]**
Gaussian Naïve Bayes	Word2Vec	0.99 [99%]	0.99 [99%]
SVM (Support Vector Machine)	Word2Vec	1.00 [100%]	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	0.98 [98%]	0.99 [99%]
SVM (Support Vector Machine)	Doc2Vec	0.99 [99%]	0.99 [99%]

Table 6.7 – F1-Measure metric using an 80/20 ratio for the Training and Testing sets respectively.

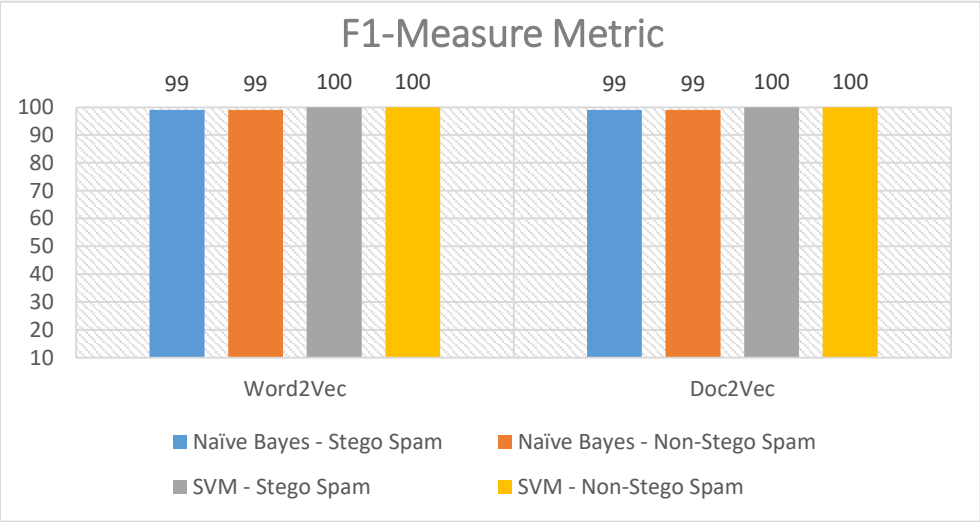


Graph 6.7 – F1-Measure metric using an 80/20 ratio for the Training and Testing sets respectively.

Table 6.7 and Graph 6.7 above demonstrate that both classifiers achieved excellent F1-Measure ratings. It was seen that the SVM classifier achieved a marginally superior F1-Measure rating compared to the Naïve Bayes classifier when implemented using the Word2Vec and the Doc2Vec algorithms. Additionally, it was seen that both classifiers achieved a lower F1-Measure rating when implemented using the Doc2Vec algorithm. For the next experiment the training and testing set sizes were adjusted to a 70/30 ratio (i.e. 1120 files for the training set and 480 files for the testing set). The results achieved and documented from these experiments are presented in Table 6.8 and Graph 6.8 below.

Independent Variables		**Dependent Variable**	
Classifier	**Feature Extraction Algorithm**	**F1-measure Result**	
		Non-stegospam [0]	**Stego Spam [1]**
Gaussian Naïve Bayes	Word2Vec	0.99 [99%]	0.99 [99%]
SVM (Support Vector Machine)	Word2Vec	1.00 [100%]	1.00 [100%]
Gaussian Naïve Bayes	Doc2Vec	0.99 [99%]	0.99 [99%]
SVM (Support Vector Machine)	Doc2Vec	1.00 [100%]	1.00 [100%]

Table 6.8 – F1-Measure metric using an 70/30 ratio for the Training and Testing sets respectively



Graph 6.8 – F1-Measure metric using an 70/30 ratio for the Training and Testing sets respectively.

By adjusting the size of the training set to 70% of the input dataset we find that the SVM classifier achieved a 100% F1-Measure rating when implemented using the Word2Vec and the Doc2Vec feature extraction algorithms, marginally better than the F1-Measure ratings it achieved in the previous experiment. Additionally, the Naïve Bayes classifier achieved the same F1-Measure ratings when implemented using the Word2Vec feature extraction algorithm as it achieved in the previous experiment. These findings are discussed further in Section 7.2 of Chapter 7. The next dependent variable tested and documented was the prediction rating achieved by each classifier implemented using the Word2Vec and the Doc2Vec feature extraction algorithms.

6.3 Classifier Prediction and Execution Time

The previous subsection presented and provided a brief discourse on the results achieved by the different dependent variables in the classification reports generated via a series of experiments. The analysis and findings thereof are deliberated in the next chapter.

This section addresses two important dependent variables namely; the prediction rate of each classifier, and the execution time of each experiment. In terms of real-world application these two metrics serve as key indicators for reliability and performance. Hence, this section presents and discusses the prediction and execution time results achieved by each classifier.

6.3.1 Classifier Predication

In machine learning, the objective of a classification module is to resolve predictive modelling problems where a class label is predicted based on a given example of input data (i.e. the testing set*)* (Bird, Klein and Loper, 2009). By splitting the input dataset into a training and testing set, we are able to evaluate the efficacy of the supervised classifiers and the NLP feature extraction algorithms implemented in this research study. The following confusion matrices illustrate the prediction results associated with each experiment conducted.

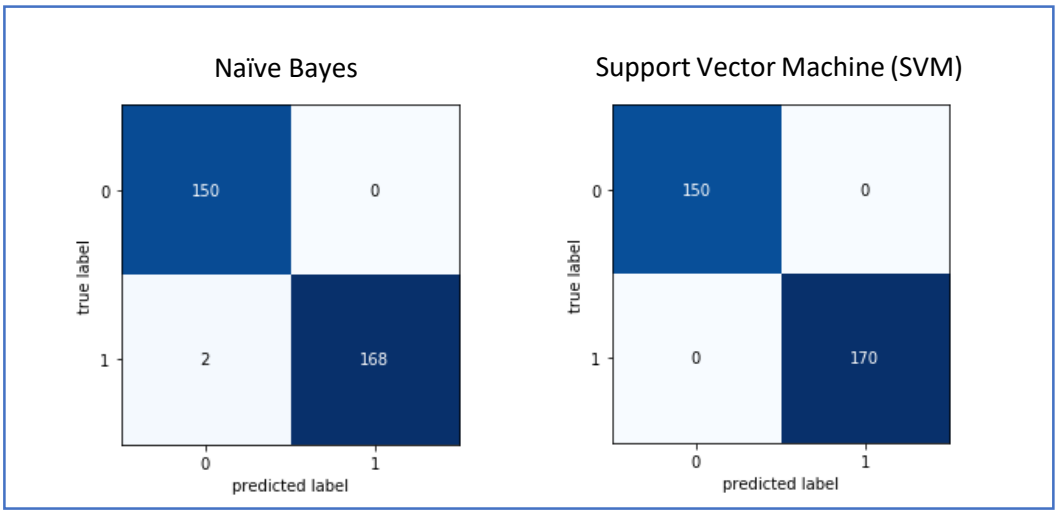
In the field of machine learning and particularly the problem of statistical classification, a confusion matrix, also known as an error matrix (Stehman, 1997), is a specific table layout that allows for the visualization of the performance of an algorithm, typically a supervised learning one. Each row of the matrix represents the occurrences in a predicted class while each column represents the occurrences in an actual class (Powers, 2011). The name originates from the fact that it makes it easy to perceive if the system is confusing two classes (i.e. classifying one as another).

- **Prediction Rates implementing Word2Vec with a 20% Testing Set**

  Table 6.9 and the confusion matrices illustrated in Graph 6.9 present the prediction results achieved by the Naïve Bayes and the Support Vector Machine classification modules implemented using the Word2Vec feature extraction algorithm with a 20/80 ratio for the testing and training sets respectively.

Classifier	NLP Feature Extraction Algorithm	Testing Set Size	Prediction
Gaussian Naive Bayes	Word2Vec	20% [320 files]	0.99375 [99.375%]
SVM	Word2Vec	20% [320 files]	1.0 [100%]

Table 6.9 – Prediction Rating achieved using Word2Vec with a 20% Testing Set



Graph 6.9 – Confusion Matrix for Naïve Bayes and SVM using Word2Vec with a 20% Testing Set

By implementing Word2Vec as the feature extraction algorithm using a 20% testing set, it was observed that the Gaussian Naïve Bayes (GNB) classifier achieved a 0.99375 [99.375%] prediction rating, whilst the SVM classifier achieved a 1.0 [100%] prediction rating as shown in Table 6.9. Additionally, this was achieved by skewing both the testing and training set by 20 files. This implies that both classifiers randomly selected 150 files labelled as non-stegospam and 170 as stegospam from the input dataset totalling to 320 files (i.e. 20% of input dataset).

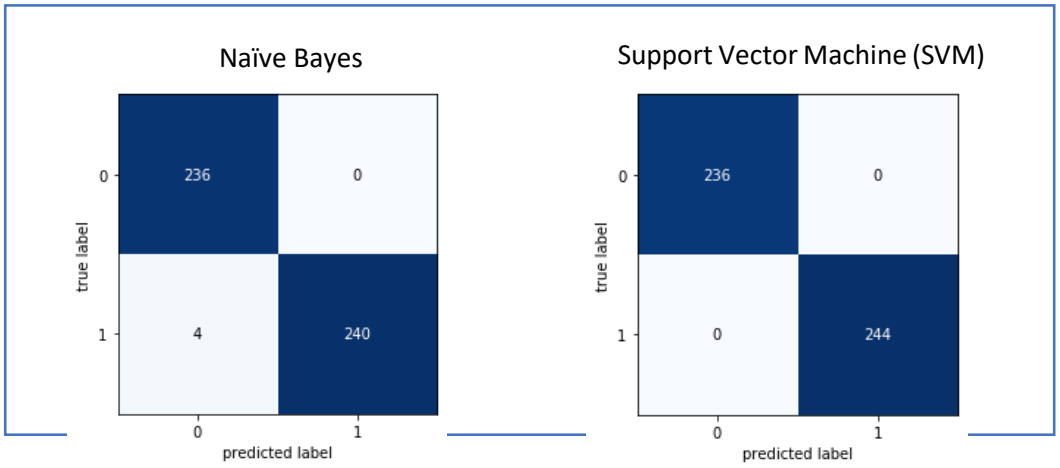
The confusion matrices illustrated in Graph 6.9, demonstrate that both classifiers were able to accurately classify stegospam files from non-stegospam files. The Naïve Bayes classifier was able to correctly classify the 150 non-stegospam files in the testing set with zero false positives. However, Naïve Bayes performed marginally lower when classifying stegospam files, correctly classifying 168 of the 170 stego spam files in the testing set, with only 2 false positives. The Support Vector Machine (SVM) classifier performed marginally better as it was able to correctly classify the 150 non-stegospam files and the 170 stego spam files in the testing set with zero false positives for both file classes as illustrated in the confusion matrices in Graph 6.9. For the next experiment, the testing and training sets were adjusted to a 30/70 ratio respectively, using the Word2Vec NLP feature extraction algorithm across both classifiers.

- **Prediction Rates implementing Word2Vec with a 30% Testing Set**

  Table 6.10 and the confusion matrices illustrated in Graph 6.10 below present the prediction results achieved by the Naïve Bayes and the Support Vector Machine classification modules implemented using the Word2Vec feature extraction algorithm with a 30/70 ratio for the testing and training sets respectively.

Classifier	NLP Feature Extraction Algorithm		Testing Set Size	Prediction
Naive Bayes	Word2Vec		30% [480 files]	0.99166 [99.166%]
SVM	Word2Vec		30% [480 files]	1.0 [100%]

Table 6.10 – Prediction Rating achieved using Word2Vec with a 30% Testing Set.



Graph 6.10 – Confusion Matrix for Naïve Bayes and SVM using Word2Vec with a 30% Testing Set.

By implementing Word2Vec as the NLP feature extraction algorithm with a 30% testing set, it was observed that the Gaussian Naïve Bayes (GNB) classifier achieved a 0.99166 [99.166%] prediction rating, whilst the SVM classifier achieved a 1.0 [100%] prediction rating as shown in Table 6.10 above. Furthermore, these predictions were achieved by skewing both the testing and training sets by 8 files implying that both classifiers randomly selected 236 files labelled as non-stegospam files and 244 as stegospam files from the input dataset totalling to 480 test files (i.e.30% of input dataset).

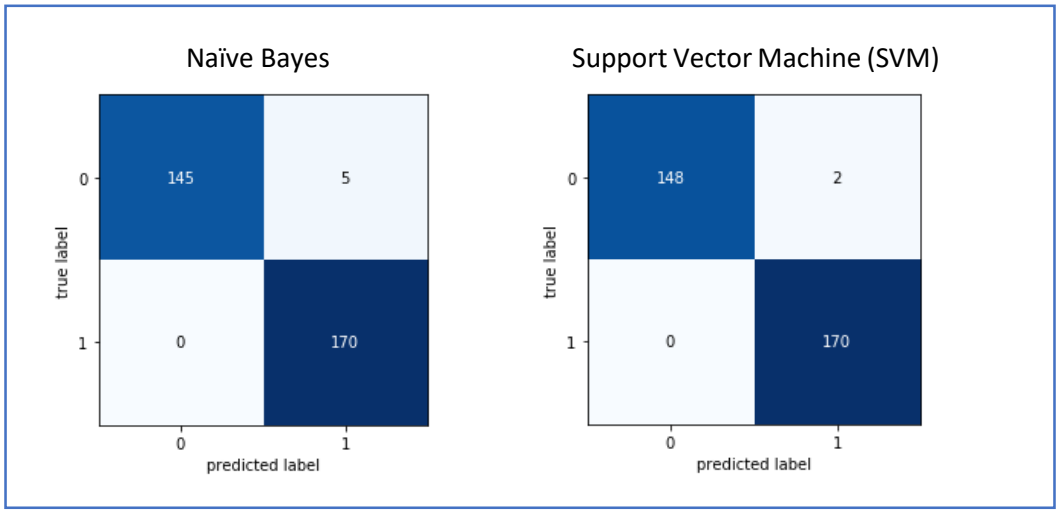
From observations of Graph 6.10, it was noticed that the Naïve Bayes classifier was able to correctly classify the 236 non-stegospam files in the testing set with zero false positives. However, it performed marginally lower when classifying the stegospam files by correctly classifying 240 of the 244 stego spam files in the testing set, with only 4 false positives. The Support Vector Machine (SVM) classifier marginally outperformed the Naïve Bayes classifier by correctly classifying the 236 non-stegospam files and 244 stego spam files in the testing set with zero false positives for both file classes as illustrated in the confusion matrices.

- **Prediction Rates implementing Doc2Vec with a 20% testing set**

  Table 6.11 and the confusion matrices illustrated in Graph 6.11 present the prediction results achieved by the Naïve Bayes and the Support Vector Machine classification modules implemented using the Doc2Vec feature extraction algorithm with a 20/80 ratio for the testing and training sets respectively.

Classifier	NLP Feature Extraction Algorithm	Testing Set Size	Prediction
Gaussian Naive Bayes	Doc2Vec	20% [320 files]	0.98437 [98.437%]
SVM	Doc2Vec	20% [320 files]	0.99375 [99.375%]

Table 6.11 – Prediction Rating achieved using Doc2Vec with a 20% Testing Set



Graph 6.11 – Confusion Matrix for Naïve Bayes and SVM using Doc2Vec with a 20% Testing Set

By implementing Doc2Vec as the NLP feature extraction algorithm with a 20% testing set, we observe that the Gaussian Naïve Bayes *(*GNB*)* classifier achieved a 0.98437 [98.437%] prediction rating, whilst the SVM classifier achieved a 0.99375 [99.375%] prediction rating as shown in Table 6.11 above. Additionally, this was achieved by skewing both the testing and training set by 20 files. This implies that both classifiers randomly selected 150 files labelled as non-stegospam and 170 as stegospam from the input dataset totalling to 320 files (i.e. 20% of input dataset).

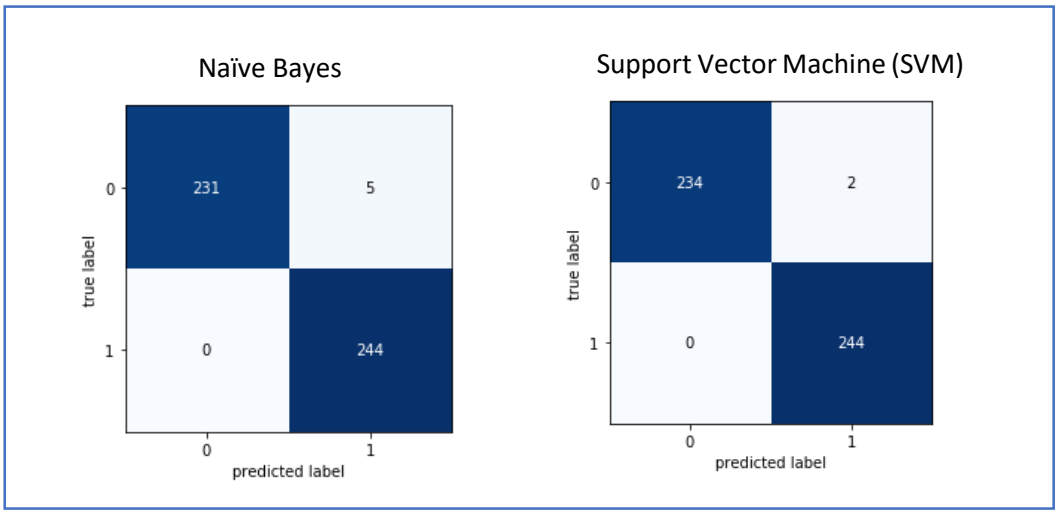
The confusion matrices shown in Graph 6.11, demonstrate that the Naïve Bayes classifier was able to correctly classify the 170 stego spam files in the testing set with zero false positives. However, it performed marginally lower when classifying non-stegospam files, correctly classifying 145 of the 150 non-stegospam files in the testing set, with 5 false positives. The Support Vector Machine (SVM*)* classifier performed marginally better than the Naïve Bayes Classifier as it was able to correctly classify the 170 stego spam files in the testing set with zero false positives and 148 of the non-stegospam files with only 2 false positives, illustrated in the confusion matrices in Graph 6.11. For the next experiment, the testing and training sets were adjusted to a 30/70 ratio respectively, implementing the Doc2Vec NLP feature extraction algorithm across both classifiers.

- **Prediction Rates implementing Doc2Vec with a 30% testing set**

  Table 6.12 below and the confusion matrices illustrated in Graph 6.12 present the prediction results achieved by the Naïve Bayes and the Support Vector Machine classification modules implemented using the Doc2Vec feature extraction algorithm with a 30/70 ratio for the testing and training sets respectively.

Classifier	NLP Feature Extraction Algorithm	Testing Set Size	Prediction
Naive Bayes	Doc2Vec	30% [480 files]	0.98958 [98.958%]
SVM	Doc2Vec	30% [480 files]	0.99583 [99.583%]

Table 6.12 – Prediction Rating achieved using Doc2Vec with a 30% testing set



Graph 6.12 – Confusion Matrix for Naïve Bayes and SVM using Doc2Vec with a 30% testing set

By implementing Doc2Vec as the NLP feature extraction algorithm using a 30% testing set, we observe that the Gaussian Naïve Bayes (GNB) classifier achieved a 0.98958 [98.958%] prediction rating, whilst the SVM classifier achieved a 0.99583 [99.583%] prediction rating as shown in Table 6.12. Additionally, these predictions were achieved by skewing both the testing and training sets by 8 files implying that both classifiers randomly selected 236 files labeled as non-stegospam files and 244 as stegospam files from the input dataset totaling to 480 test files (i.e.30% of input dataset).

Observing the confusion matrices for the experiment illustrated in Graph 6.12, it can be seen that the Naïve Bayes classifier was able to correctly classify 231 of the 236 non-stegospam files in the testing set with 5 false positives. However, Naïve Bayes performed much better when predicting stegospam files by correctly classifying 244 of the 244 stego spam files in the testing set, with zero false positives. The Support Vector Machine (SVM) classifier marginally outperformed the Naïve Bayes Classifier when predicting non-stegospam files as it was able to correctly classify 234 of the 236 non-stegospam files with only 2 false positives and 244 of the 244 stego spam files in the testing set with zero false positives as illustrated in the confusion matrices.

- **Summary of the Predication Rating**

NLP Feature Extraction Algorithm	Testing Set Size	Naïve Bayes Prediction Rating	SVM Prediction Rating
Word2Vec	20%	99.375%	100%
Word2Vec	30%	99.166%	100%
Doc2Vec	20%	98.437%	99.375%
Doc2Vec	30%	98.958%	99.583%

Table 6.13 – Summary of the Prediction Ratings achieved

Examining Table 6.13 above, it can be seen that the Support Vector Machine (SVM) classification module achieves a 100% prediction when implemented using the Word2Vec feature extraction algorithm for both set size ratios (i.e. 20% and 30%). SVM marginally outperformed the Naïve Bayes classification module which achieved a prediction rating between 99.166% to 99.375% when implemented using Word2Vec.

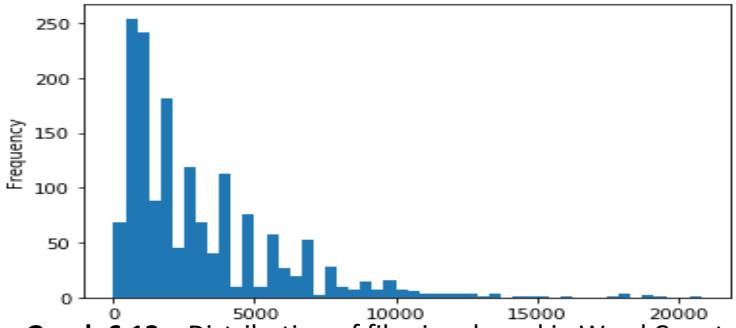
When implemented using the Doc2Vec feature extraction algorithm, it can be found that both classifiers achieve lower prediction ratings when compared to the prediction ratings achieved using the Word2Vec algorithm. However, these ratings improved when the test set size was adjusted to 30% of the input dataset. This infers that by increasing the test set size, Doc2Vec will achieve better prediction ratings.

Conversely, it was found that the prediction rating of the Naïve Bayes classifier decreased when the test set size was adjusted to 30%. This may be attributed to the manner in which Word2Vec establishes vectors and mines its features when compared to Doc2Vec. Word2vec and Doc2Vec were discussed in greater detail in Chapter 4 (Natural Language Processing).

6.3.2 Execution Time for each experiment

Once the prediction results achieved by each experiment through the implementation of the different NLP feature extraction algorithms, test set sizes and classifiers have been recorded and reviewed, the next dependent variable to be considered was the time lapse or execution time of each experiment conducted. As mentioned in the preceding chapter, the input dataset used for this research study comprised of a diverse size of spam files for both file

classes. This strategy was employed to assess the performance and constancy of the proposed steganalysis. Graph 6.13 below illustrates the distribution of the file sizes contained within the dataset based on its word count. The word count per file plotted on the X-axis range from 0 to over 20,000 words, with a higher frequency of files consisting of 5000 or less words. Overall, the dataset used to conduct each experiment comprised of 814,077 words, 67,698 sentences and 6,418 paragraphs. Evaluating the causal effect induced by the independent variables on the performance of each experiment is critical in ascertaining the efficiency of the constructed prototype if integrated into a real-world solution.



Graph 6.13 – Distribution of file sizes based in Word Count

- **Execution Time – Naïve Bayes Classifier**

	Experiment Duration – Naïve Bayes (NB) Classifier Measurement- Minute: Seconds:100th/Sec	
Test Set Size	**Word2Vec**	**Doc2Vec**
20%	07:58:51	08:18:29
30%	07:52:32	12:47:41

Table 6.14 – Execution Time – Naïve Bayes Classifier

Table 6.14 above tabulates the execution time for all experiments conducted using the Naïve Bayes classifier with the different test set sizes and feature extraction algorithms. Based on the results recorded we find that Naïve Bayes has a better execution time when implemented using Word2Vec when compared to Doc2Vec.

When implemented using the Doc2Vec feature extraction algorithm Naïve Bayes performed much slower, which is further highlighted when the test set size was increased to 30% of the input dataset, achieving an execution time of 12:47:41 (04:55:09s slower

than Word2Vec). This may be attributed to the fundamental manner in which Doc2Vec constructs its vectors. Doc2vec builds its feature sets using paragraph vectors, which are unique among all documents within the same dataset and therefore takes longer to build than Word2Vec. Word2Vec in contrast generates word vectors which can be shared or learnt from another document within the same dataset, consequently improving its execution time (Ma, 2018).

- **Execution Time – Support Vector Machine (SVM)**

	Experiment Duration – Support Vector Machine (SVM) Classifier Measurement- Minute: Seconds:100th/Sec	
Test Set Size	**Word2Vec**	**Doc2Vec**
20%	07:59:53	08:18:39
30%	07:53:13	12:48:59

Table 6.15 – Execution Time – Support Vector Machine (SVM)

Table 6.15 above tabulates the execution time for all experiments conducted using the Support Vector Machine (SVM) classifier with the different test set sizes and feature extraction algorithms. Based on the results recorded we find that Word2Vec once again outperforms the Doc2Vec feature extraction algorithm. When implemented using the Doc2Vec feature extraction algorithm SVM performed much slower, which is further highlighted when the test set size was increased to 30% of the input dataset, achieving an execution time of 12:48:59 (04:55:46s slower than Word2Vec). Once again it may be attributed to the fundamental manner in which Doc2Vec constructs its vectors.

- **Summary of the Execution Time**

In studying the execution time of each experiment it can be acknowledged the direct impact that each independent variable has on the performance of the prototype or steganalysis process. This may be attributed to several underlying factors and procedures which are specific in nature to each classifier or feature extraction algorithm. Chapter 4 of this study delivered a comprehensive discourse on the structure and processes employed by the classifiers and feature extraction algorithms implemented by this study. Whilst performance is an important metric, it must be

sacrificed for accuracy and precision. Further discussions related to the performance of the steganalysis is presented in Chapter 7.

6.4 Chapter Conclusion

The results of the experiments conducted are presented in this chapter. The first section provides a discussion on the observations and measurements of the quantitative data recorded from each experiment. Section 6.2 presents the classification results of each experiment in terms of the dependent variables measured, as discussed in the methodology chapter and section 6.1 of this chapter. Finally, section 6.3 presents and discusses the prediction ratings and execution times recorded for each experiment. The next chapter presents and discusses the research findings.

7

Findings, Conclusions and Future Work

This chapter provides a summary of this research's contributions. Following which, the author reflects on the results and provides some conclusions. Finally, a few recommendations for future work are presented.

This chapter presents the findings, conclusion and recommended future work for this research. The primary objective of this research was to evaluate the hypothesis by testing the efficacy of a steganalysis approach that integrated Natural Language Processing methods to successfully classify stegospam files created using a probabilistic context-free grammar from non-stegospam files.

Chapter 1 provided an introduction to the research problem and questions that this research study set out to answer. Chapter 2 provided a discussion on the email system's infrastructure, spam email and email vulnerabilities. Chapter 3 discussed the different types of steganography based on the different cover mediums with a focus on text-based steganography. Chapter 4 introduced and deliberated on the various Natural Language Processing techniques that were considered for the proposed solution. Chapter 5 presented the research design and methodology followed by this research and described the experiments that were performed for this study. Chapter 6 presented and discussed the results recorded for the research. This chapter presents the findings, conclusions and recommended future work for this research study.

The structure of this chapter is as follows:

Section 7.1 - Introduction on the research findings and metrics recorded

Section 7.2 – Discussion and Summary on the Research Findings

Section 7.3 – Conclusions

Section 7.4 – Research Contribution

Section 7.5 – Future Work

Section 7.6 – Conclusion

7.1 Introduction

The previous chapter presented and discussed the experimental data generated by the different dependent variables based on the vicissitudes to the independent variables. This data was then used in comparative analysis to ascertain the efficacy of the different NLP feature extraction and classification or machine learning methods to accurately classify stego from non-stegospam files. In addition, the execution time of each experiment was observed and documented to evaluate any performance fluctuations directly influenced by changes to the independent variables.

In this chapter the evidence produced are analysed, and the findings thereof are presented. This is followed by the conclusions and recommended future work.

7.2 Discussion and Summary on the Research Findings

An experimental research approach was elected to assess the ability and limitations of the different natural language processing methods integrated into the proposed steganalysis. Experiments comprised of a set of tests to measure the ability of each feature extraction and classification method. The objective of implementing different feature extraction algorithms and classification methods was to discover how successful each were in detecting stego from non-stegospam files. Additionally, it was also to provide a mechanism for repeatability when performing the experiments.

Based on the findings of this research, it can be determined whether the implementation of natural language processing methods in the proposed steganalysis can successfully classify stegospam from non-stegospam files and which feature extraction and classification methods achieved the best results subsequently, addressing the research question and relevant secondary research questions.

This section presents the findings of the empirical evidence gathered through a succession of experiments as outlined in Chapter 6 – Experimental Results based on the different dependent variables namely; Accuracy, Precision, Recall, F1-Measure, Prediction and Execution time. These are discussed as follows:

- **Accuracy -** measures the inputs or files in the test set that a classifier correctly classifies or labels (Bird, Klein and Loper, 2009). Consequently, the empirical evidence recorded in Table 6.1 and Table 6.2 of Chapter 6 demonstrate that overall both the Naïve Bayes (NB) and the Support Vector Machine (SVM) classifiers achieved excellent accuracy ratings. Additionally, it was discovered that when implemented using the Word2Vec feature extraction algorithm, both classifiers achieved better accuracy ratings compared to when they were implemented with the Doc2Vec feature extraction algorithm. This may be fundamentally attributed to the manner in which each feature extraction algorithm establishes its feature sets – which were discussed in greater detail in Chapter 4 (Natural Language Processing).

  Additionally, it was found that when the training and testing set ratios are adjusted, the accuracy ratings of both classifiers (when implemented using Word2Vec or Doc2Vec) remain unchanged. Only Naïve Bayes recorded a marginal improvement when implemented using Doc2Vec. This demonstrates the reliability of the proposed steganalysis.

- **Precision** refers to the relation of currently predicted positive observations against the total predicted positive observations (i.e. TP + FP) (Miyasato, 2020). Based on the empirical evidence recorded in Table 6.3 and Table 6.4 of Chapter 6, it was found that both classifiers (i.e. Naïve Bayes and Support Vector Machine) achieved excellent precision ratings when classifying stegospam from non-stegospam files. Overall, both classifiers achieved better precision ratings when implemented using the Word2Vec feature extraction algorithm as opposed to the Doc2Vec algorithm. Additionally, by adjusting the training set to 70% and the testing set to 30%, it was found that the precision ratings of the Naïve Bayes classifier implementation using Doc2Vec marginally improved. This reveals that the Naïve Bayes classifier produces better precision ratings when implemented with the Doc2Vec feature extraction algorithm at a 70/30 ratio. The SVM remained unchanged with vicissitudes to the training and testing sets when implemented for both Word2Vec and Doc2Vec. Furthermore, the

results produced with fluctuations to the training and testing sets demonstrates the reliability of the proposed steganalysis.

- **Recall** is the ratio of correctly predicted positive observations of all observations in the actual class labelled positive (Bird, Klein and Loper, 2009; Miyasato, 2020). Analysing the empirical evidence recorded in Table 6.5 and Table 6.6 of Chapter 6, it was found that the Naïve Bayes (NB) and the Support Vector Machine (SVM) classifiers achieved better Recall ratings when implemented with the Word2Vec algorithm compared to implementation with the Doc2Vec algorithm.

  Word2Vec, SVM implementation achieved a 100% Recall rating when classifying both stegospam and non-stegospam files. The same Recall rating was achieved by SVM when the training and testing set sizes were adjusted. Naïve Bayes, on the other hand, achieved a 99% Recall rating when detecting stegospam files and 100% for non-stegospam files. This rating decreased to 98% when the testing set was adjusted to 30% of the input dataset.

  Implementation with Doc2Vec, Naïve Bayes achieved a marginally lower Recall rating than the SVM classifier. However, by adjusting the training and testing set sizes to 70/30 respectively, it was found that the Recall rating for the Naïve Bayes classifier improved. This, therefore, demonstrates that the Naïve Bayes classifier performs marginally better when implemented with the Doc2Vec feature extraction algorithm when there are certain changes to the training and testing set sizes. Overall both classifiers achieved excellent Recall ratings across all experiments.

- **F1-Measure or F1-Score** combines the precision and recall to produce a single weighted average, defined as the harmonic mean of the precision and recall. Consequently, this score includes both false positives and false negatives (McIntyre, Bengfort and Gray, 2019). Observing the F1-Measure ratings recorded in Table 6.7 and Table 6.8 of Chapter 6 it was found that both classifiers achieved better results when implemented using the Word2Vec algorithm as opposed to the Doc2Vec algorithm.

By adjusting the testing and training set sizes it was discovered that the F1-Measure ratings remained the same for both classifiers when implemented using the Word2Vec algorithm. However, when implemented using the Doc2Vec algorithm the F1-Measure ratings for both classifiers improved.

- **Prediction.** In machine learning, the objective of a classification module is to resolve predictive modelling problems where a class label is predicted based on a given example of input data (i.e. the testing set) (Bird, Klein and Loper, 2009). Observing Tables 6.9 – 6.13 of Chapter 6, it can be seen that overall, the Support Vector Machine classifier achieved marginally better prediction rates when compared to the Naïve Bayes classifier, which achieved a prediction rate ranging from 99.375% to 100%. The prediction ratings achieved by the Naïve Bayes classifier ranged from 99.166% - 99.583% across all experiments.

- **Execution Time** refers to the time that the constructed prototype takes to pre-process the input dataset, mine features to build a feature set and to train and test the machine learning module in order to make a prediction. Observing Table 6.14 and Table 6.15 of Chapter 6 it was discovered that the constructed prototype performed most efficiently when implemented using Word2Vec, Naïve Bayes and a 30% test set size achieving an execution time of 7 minutes, 52 seconds and 32 hundredths of a second [07:52:32]. Other combinations of the dependent variables yield marginally slower times with the implementation of the Doc2Vec algorithm producing the slowest times.

- **Summary of the Findings**

  The empirical evidence recorded suggests that overall, the implementation of the Word2Vec feature extraction algorithm produces better results as opposed to the implementation of the Doc2Vec algorithm. Conversely, this may be accredited to the manner in which Doc2Vec mines and establishes a feature set. Doc2Vec creates a vectorised representation of a collection of words, regarding it as a single unit. It doesn't just provide a simple average of the words in the sentence, instead it builds deeper and more meaningful word embeddings hence, taking longer to construct the

feature set (Nicholson, 2019). Doc2Vec and Word2Vec were discussed in greater detail in Chapter 4 (Natural Language Processing).

Additionally, based on the experimental evidence the Support Vector Machine achieved marginally better results and execution times when compared to the Naïve Bayes classifier. Lastly, the evidence produced suggests that if the size of the input dataset and testing sets were to be increased the dependent variable ratings would improve.

- The first objective (OB1) was to: **identify and discuss the vulnerabilities and motivation for the election of spam email as a covert channel.** Based on the discussion and review of the email systems in particular spam email presented in Chapter 2 we were able to successfully identify and discuss numerous intrinsic vulnerabilities and motivational factors that strongly advocate the election of spam email as an ideal medium for covert based communications.

- The second objective (OB2) of this research study was to: **identify existing linguistic steganalysis methods and to determine its suitability for the detection of stego spam files generated using a probabilistic context-free grammar (PCFG) from non-stegospam files.** This research study satisfies this objective through the identification and discussion of existing linguistic steganalysis approaches presented in Chapter 3, Section 3.6 which demonstrates their inadequacy to efficaciously detect stego spam files generated using a PCFG. Existing linguistic steganalysis approaches fundamentally focus on the detection of the stego-text created using synonym substitution - a common lexical steganographic approach. Additionally, the methods employed by these approaches require the original cover text to discover any statistical or lexical deviations between the two files, or to challenge the context fitness of a synonym within each sentence.

- The third objective (OB3) was to: **identify and describe various natural language processing (NLP) methods or algorithms that may be considered for the proposed steganalysis.** Based on the discussion and review of the different

NLP algorithms and methods presented in Chapter 4 (Natural Language Processing) several key data pre-processing methods were successfully identified (such as, Tokenization, Lemmatization, Stop Word Removal, etc.), feature extraction (i.e. Word2Vec and Doc2Vec), and classification methods (Naïve Bayes and SVM) to be considered and implemented in the proposed steganalysis.

- The fourth objective (OB4) was to: **evaluate the ability of a proposed NLP-based steganalysis by conducting several controlled experiments on existing corpora known to contain stego and non-stegospam files.** This research study was able to achieve this objective through the design and development of an NLP-based prototype, the establishment of corpora comprising of stego and non-stegospam files, and the establishment of a test framework for conducting controlled experiments as described in Chapter 5 (Research Design and Methodology). The test framework comprised of experiments that measured the efficacy and performance of the task required to classify stego from non-stegospam files. The results of this research were recorded and presented in Chapter 6 (Experimental Results).

- The final objective (OB5) was to: **identify any limitations not addressed by the existing or the proposed steganalysis, which can be addressed in further research**. This research study achieved this objective by identifying the limitations associated with existing linguistic steganalysis as discussed in Chapter 3, the limitations of the proposed steganalysis discussed in Chapter 5, Section 5.5 and Section 7.4 of this chapter which recommends the future work for this research study.

7.3 Conclusions

This section addresses the research question and the hypothesis of this research study.

7.3.1 Conclusion on the Research Questions

Main Research Question:

Can the implementation of Natural Language Processing methods in a steganalysis approach effectively detect or classify stegospam files created using a probabilistic context free grammar from non-stegospam files?

Indeed, this research clearly demonstrated that the implementation of NLP methods in a steganalysis approach can effectively detect stegospam from non-stegospam files as discovered in the experimental results presented in Chapter 6 (Experimental Results). Despite the different amalgamations of feature extraction algorithms and classification modules producing marginally better results when compared to one another, overall, they achieved excellent Precision, Accuracy, Recall, F1-Measure and Prediction results of no less than 97%, underlining the efficacy and reliability of the proposed NLP-based steganalysis. In addition, the steganalysis returned excellent execution times considering it had to process and analyze 67,698 sentences per experiment.

SRQ1. Why is spam email considered an ideal carrier for covert communications?

This research study adequately addressed this question in Chapter 2 by presenting and analysing the structure of the email system; its inherit lack of security; and the various factors that contribute to the election of email spam as an ideal carrier for covert communications. These include the ubiquitous nature of spam; its presumed innocuity; and its affordability. In addition, Section 2.8 of Chapter 2 highlights known cases where email spam was employed as a covert communication channel to perpetrate crimes.

SRQ2. Which linguistic steganographic techniques are favoured in order to embed a secret message within plaintext files? This research study addresses this question by presenting and discussing existing linguistic steganographic techniques employed by steganographers to conceal a message which included Synonym Substitution (SS), Syntactic and Semantic

transformation techniques using Block-Code, Mixed-Radix Number or the Huffman Code algorithms as discussed in Chapter 3, Section 3.4 and Section 3.5.

SRQ3. Which steganalysis techniques are commonly employed for the detection of linguistic steganography? This research study addresses this research question by presenting and reviewing the different steganalysis approaches or techniques used to detect linguistic steganography as presented and discussed in Chapter 3, Section 3.6.

SRQ4. Which Natural Language Processing methods or algorithms are most effective in detecting or classifying stego from non-stegospam files? Based on the research design, the elected methodology and the series of controlled experiments this research study undoubtedly answered this key question. From the empirical evidence and the research findings documented in Chapter 6, this study can conclude that the Naïve Bayes and the Support Vector Machine *(*SVM*)* classification methods achieved better Accuracy, Precision, Recall, F1-Measure and Predication results when implemented using the Word2Vec feature extraction algorithm as opposed to the Doc2Vec algorithm. Additionally, the Support Vector Machine achieved better results than the Naïve Bayes classification method.

7.3.2 Conclusion on the Research Hypothesis

To test the hypothesis of this research study an experimental approach was adopted, as presented in the research design and methodology chapter. The designed experiments tested the capabilities of the proposed NLP-based steganalysis via the implementation of a constructed prototype by measuring the following metrics:

- Accuracy
- Precision
- Recall
- F1-Measure
- Predication rates.

These metrics are measured and assessed based on the vicissitudes to the independent variables, namely; the size of the training and testing sets, the implementation of the different feature extraction algorithms and the machine learning or classification method. From the research results and research findings chapters it can be concluded that the implementation

of Natural Language Processing methods veritably effectively detects or classifies non-stego from stego spam files generated using a probabilistic context-free grammar.

The section discussed the research conclusions for the research questions and hypothesis. The next section discusses the contributions of this research.

7.4 Research Contributions

The previous section presents the conclusions of this research, this section discusses the contribution that this research has to the field of linguistic steganalysis.

This research study highlights and analyses the current state, capabilities and the drawbacks associated with existing linguistic steganalysis approaches and provides insight and knowledge into their inability to effectively classify stegospam from non-stegospam files generated using a PCFG.

Consequently, this research offers an innovative and alternative approach to linguistic steganalysis, one that encapsulates Natural Language Processing and Machine Learning methods and algorithms to deconstruct and analyse volumes of data, and also one that constructs deep mathematical and statistical relationships between the various words, sentences and paragraphs to classify a body of text. In doing so, this research provides knowledge and insight to the capabilities of the proposed steganalysis. Furthermore, the anatomy and design of the proposed steganalysis allows it to be adaptable and repeatable in different situations underpinning the positivistic paradigm.

Additionally, in Chapter 5 (Research Design and Methodology) the constructed prototype was simply identified as an artefact implemented to test the proposed steganalysis. However, the results achieved through the prototype provide useful knowledge and insight when making decisions on the development of a software security solution that is considering which Natural Language Processing and Classification methods and algorithms to implement.

This section describes the contribution of this research in relation to the research questions that have been presented. The next section discusses the recommended future work.

7.5 Recommended Future Work

The contribution of this research study highlighted the knowledge gained by undertaking this research.

Future work on the linguistic steganographic approach presented in this research study can lead to a better understanding on how to improve its performance and capabilities. Implementing Natural Language Processing and Machine Learning methods in the field of cyber-security can provide exceptional benefits in the early detection or mitigation of potential threats. Text-based communication is the *'lifeline'* of the modern era and social media is one of its prime sources, fostering ideal opportunities for covert channels. Consequently, further research and adaptability of the proposed steganalysis could be implemented to scrutinise and classify different data types from various sources.

One of the limitations of this research is the speed of execution. Considering that Word2Vec and Doc2Vec are neural-based feature extraction algorithms which take longer to mine features from a corpus, future research of different algorithms such as Fast*Text*, Chainer or Seq2Seq may achieve better results and execution times.

From an input dataset perspective, the data points included in dataset for this research only contained the body or contents of each spam file type. Future research could include the metadata and attachments of email files. Furthermore, the stego spam objects were generated using only Spammimic as it best mimicked spam emails. Therefore, future research could be done to assess and provide insight into the capability and adaptability of the proposed steganalysis in classifying stego objects generated using other linguistic steganographic tools such as Texto and T-Lex.

7.6 Chapter Conclusion

This chapter provides the conclusion to this research study which investigates the effectiveness of Natural Language Processing methods in classifying stego spam files generated using a PCFG from non-stegospam files.

The overview of the research features a discussion on the motivation and opportunities that spam email presents as a covert channel. Additionally, it provides an overview of the different forms of steganography with emphasis on linguistic steganography and the limitations associated with current linguistic steganalysis approaches. The research also presents a detailed discussion on the various natural language processing methods and how they address such limitations.

A summary of the research is presented to highlight the discussions of the previous chapters and how the research had been conducted. In conclusion, the contribution of the research is presented, followed by the recommended future research that could be conducted.

Based on the results achieved and presented in this research, the deduction is that natural language processing methods can successfully classify stego spam files generated using a PCFG from non-stegospam files, therefore, subsequently addressing the limitations of existing linguistic steganalysis approaches.

Bibliography

Abdul-Mahdi, N.H., Yahya, A. and Ahmad, R.B., 2013. Secure and robust information hiding scheme. *Procedia Engineering*, Volume 53, pp. 463 -471.

Ahvanooey, M.T., Li, Q., Hou, J., Rajput, A.R. and Yini, C.M.T., 2019. Modern text hiding, text steganalysis, and applications: a comparative analysis. *Entropy*, pp. 350 -381.

Adam, M., 2018. Not your fairy-tale prince: The Nigerian business email compromise threat. *Computer Fraud and Security*, August, pp. 14-16.

Agarwal, M., 2013. Text steganography approaches: a comparison. *International Journal of Network Security and Its Application* (IJNSA), 5(1).

Agirre, E. and Edmonds, P., 2006. *Word Sense Disambiguation: Algorithms and Applications*. 1 ed. Netherlands: Springer.

Alasadi, S., 2017. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, Volume 12, pp. 4102-4107.

Ali, Z., 2019. *Simple tutorial on word embedding and Word2Vec*. [Online]
Available at <https://medium.com/@zafaralibagh6/simple-tutorial-on-word-embedding-and-word2vec-43d477624b6d> [Accessed 11 September 2020].

Alkahtani, H., Gardner-Stephen, P. and Goodwin, R., 2019. *A taxonomy of Email SPAM Filters*. College of Computer Science and Information Technology, King Faisal University, pp. 356-362.

Almohammad, A., 2010. *Steganography-based secret and reliable communications: Improving Steganographic Capacity and Imperceptibility*. Uxbridge, West London, UK: Brunel University.

Alyafei, D.M., 2018. *Natural language analysis*. [Online]
Available at: http://www.contrib.andrew.cmu.edu/~dyafei/NLP.html [Accessed 5 October 2019].

Atallah, M.J., McDonough, C.J. and Raskin, V., 2001. *Natural Language Processing for Information Assurance and Security: An Overview and Implementations*. Proceedings of the 2000 workshop on New security paradigms, pp. 51-65.

Bajaj, K. S. and Pieprzyk, J., 2013. Can we CAN the Email Spam. *Proceedings of the 2013 Fourth Cybercrime and Trustworthy Computing Workshop,* pp. 36-48.

Bajpai, S. and Saxena, K., 2013. Enhancement of Security and Embedding capacity through Huffman Coding in Steganography. *International Journal of Emerging Trends & Technology in Computer Science* (IJETTCS), 2(4), pp. 73-78.

Bambrick, N., 2016. *KDNuggets : Support Vector Machines - What are they?.* [Online]
Available at: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html [Accessed 19 August 2019].

Bengio, Y., 2008. *Scholarpedia - Neural net language models*. [Online]
Available at: http://www.scholarpedia.org/article/Neural_net_language_models
[Accessed 15 August 2019].

Bhat, D., Krithi, V., Manjunath, KN., Prabhu, S. and Renuka, A., 2017. Information hiding through dynamic text steganography and cryptography. *IEEE*, pp. 1826-1830.

Bhowmick, A. and Hazarika, S.M., 2016. *Machine learning for e-mail spam filtering: review techniques and trends.* Assam, India: Tezpur University.

Bird, S., Klein, E. and Loper, E., 2009. *Natural language processing with Python*. Sebastopol: O'Reilly Media.

Brush, K. and Ferguson, K., 2019. *WhatIs.com*. [Online]
Available at: https://whatis.techtarget.com/definition/e-mail-electronic-mail-or-email
[Accessed 20 September 2020].

Bujang, R.Y. and Hussin, H., 2013. Should We Be Concerned with Spam Email? A look at its Impacts and Implications. Rabat, *IEEE*, pp. 1-6.

Carnie, A., 2013. *Syntax: A generative introduction*. 3rd ed. Malden, MA: Wiley-Blakwell.

Carr, W., 2006. Philosophy, Methodology and Action Research. *Journal of Philosophy of Education.* 40(4), pp. 421-435.

Castiglione, A., Alfredo, D.S., Fiore, U. and Palmieri, F., 2012. An asynchronous covert channel using spam. *Computers and Mathematics with Applications*, 63, pp. 437-447.

Chandel, A., Aggarwal, A., Mittal, A. & Choudhury, T., 2019. *Comparative Analysis of AES & RSA Cryptographic Techniques.* Dubai, United Arab Emirates, 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE).

Chang, C.Y. and Clark, S., 2014. Practical Linguistic Steganography using contextual synonym substitution and a Novel Vertex Coding Method. *Association for Computational Linguistics*, 40(2), pp. 403-442.

Changder, A. and Majumder, S., 2013. *A novel approach for text steganography: generating Text Summary using Reflection Symmetry*. International Conference on Computational Intelligence, 10, pp. 112 - 120.

Chang, K.W., 2016. *Dependency parsing*, Virginia: University of Virginia.

Chaudhary, S., Dave, M. and Sanghi, A., 2016. Review of linguistic text steganographic methods. *International Journal on Recent and Innovation Trends in Computing and Communications,* 4(7), pp. 377-381.

Chomsky, N., 1957. *Syntactic Structures.* 1st ed. Berlin: Mouton De Gruyter.

Chowdhury, G., 2003. Natural Language processing. *Annual Review of Information Science and Technology,* 37, pp. 51-89.

Chow, P., 2012. Surfing the Web anonymously - The good and evil of the anonymizer. *SANS Institute,* 9 July.

Christina, V., Karpagavalli, S. & Suganya, G., 2010. A Study on Email Spam Filtering Techniques. *International Journal of Computer Applications,* 12(1), pp. 7-9.

Christensson, P., 2006. TechTerms. [Online]
Available at < https://techterms.com/definition/pixel> (Accessed 20 October 2020).

Chuan, C.H., 2009. HTTP - Security with SSL. [Online]
Available at: https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_SSL.html [Accessed 25 July 2019].

Clark, J., Ruoti, S., Seamons, K., Zappala, D. and Van Oorschot, P.C., 2018. Securing Email. pp. 1-14.

Clark, K. and Manning, C.D., 2016. *Deep reinforcement learning for mention-ranking coreference models.* Computer Science Department - Stanford University.

Clement, J., 2020. *Statista: Spam: share of global email traffic 2014-2020*. [Online]
Available at < https://www.statista.com/statistics/420391/spam-email-traffic-share/> [Accessed 20 August 2020].

Cole, E., 2003. *Hiding in plain sight -steganography and the art of covert communication*. Indianapolis: Wiley Publishers.

Cormack, G.V., 2008. Email spam filtering: A Systematic Review. *Foundations and Trends in Information Retrieval*, 1(4), pp. 1-42.

Cormack, G.V. and Lynam, T.R., 2006. TREC 2005 Spam Track Overview. [Online]
Available at < http://plg.uwaterloo.ca/gvcormac/trecspamtrack05> [Accessed 23 11 2018].

Cothenet, C., 2019. *Dependency Parser or how to find syntactic neighbours of a word. [Online] Available at < https://towardsdatascience.com/dependency-parser-or-how-to-find-syntactic-neighbours-of-a-word-a9e7d17ffe8>* [Accessed 8 November 2020].

Craig, T., 2012. *NLP-driven ontology modelling*. [Online]
Available at
<https://www.ibm.com/developerworks/community/blogs/nlp/entry/nlp_driven_ontology_modeling8?lang=en> [Accessed 30 September 2019].

Creswell, J.W., 1994. *Research design: qualitative and quantitative approaches*. Thousand Oaks, CA, SAGE Publications.

Crosson, A., 2016. *Summarize Documents using TF-IDF*. [Online]
Available at <https://medium.com/@acrosson/summarize-documents-using-tf-idf-bdee8f60b71> [Accessed 15 August 2019].

Dan, V., 2018. Empirical and Non-Empirical Methods. In: J. Matthes, ed. *The International Encyclopaedia of Communication Research Methods.* Berlin: Free University of Berlin, Germany.

Decarlo, M., 2018. *Scientific Inquiry in Social Work.* Roanoke, VA: Creative Commons Attribution-Non-Commercial-ShareAlike 4.0 International License.

Denise F. and Polit, C.T.B., 2004. *Nursing Research: Principles and Methods*. 7th ed. New York: Lippincott Williams & Wilkins.

Dennis, A.R. and Valacich, J.S., 2001. Conducting experimental research in information systems. *Communications of the Association for Information Systems,* 7(5), pp. 1 - 41.

Din, R., Samsudin, A. and Lertkrai, P., 2012. A framework components for natural language Steganalysis. *International Journal of Computer Theory and Engineering*, 4(4), pp. 641-645.

Dormer, L., 2020. *Email marketing trends 2020*. [Online]
Available at <https://www.smartinsights.com/email-marketing/email-marketing-trends-2020> [Accessed 26 September 2020].

Dorr, B., Zajic, D. and Schwartz, R., 2003. Cross-Language Headline Generation for Hindi. *ACM Transactions on Asian Language Information Processing (TALIP),* 2(3), pp. 270-289.

Dubrovin, V., 2017. *Myths and legends of SPF*. [Online]
Available at <https://hackernoon.com/myths-and-legends-of-spf-d17919a9e817>
[Accessed 30 September 2020].

Duffy, G., 2008. Chapter 11 – Pragmatic analysis. In: *Qualitative Methods in International Relations.* Basingstoke, United Kingdom: Palgrave MacMillan, pp. 168 - 169.

Edgar, T.W. and Manz, D.O., 2017. *Research Methods for Cyber Security*. Cambridge: Elsevier.

Elsadig, M.E., Kiah, L.M., Zaidan, B.B. and Zaidan, A.A., 2009. *High Rate Video Streaming Steganography*. International Conference of Future Computer and Communication, pp. 672-675.

Enge, E., 2015. *IDF and the Importance of Uniqueness.* [Online]
Available at < https://moz.com/blog> [Accessed 20 October 2020].

Espanol, C., 2017. Medium.com - What are the different levels of NLP? How do these integrate with Information Retrieval? [Online]
Available at <https://medium.com/@CKEspanol/what-are-the-different-levels-of-nlp-how-do-these-integrate-with-information-retrieval-c0de6b9ebf61> [Accessed 2 October 2019].

Fan, X., 2010. *A study of attacks on collaborative spam filtering*. Athens, Georgia: Central South University, China.

Fellbaum, C., 2006. Princeton University: WordNet - A lexical database for English. [Online]
Available at < https://wordnet.princeton.edu/> [Accessed 22 June 2020].

Fletcher, T., 2009. *Support Vector Machines Explained*, London: University College London (UCL).

Fortney, K., 2017. *TowardsDataScience: Pre-Processing in Natural Language Machine Learning.* [Online]
Available at < https://towardsdatascience.com/pre-processing-in-natural-language-machine-learning-898a84b8bd47> [Accessed 23 October 2020].

Gangavarapu, T., Jaidhar, C. and Chanduka, B., 2020. Applicability of machine learning in spam and phishing email filtering: review and approaches. *Artificial Intelligence Review*, 53.

Garbade, M.J., 2018. *A quick introduction to text summarization in machine learning*. [Online]
Available at < https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f> [Accessed 9 September 2019].

Geitgey, A., 2018. *Natural Language Processing is fun!* [Online]
Available at <https://medium.com/@ageitgey/text-classification-is-your-new-secret-weapon-7ca4fad15788> [Accessed 8 October 2019].

Gibbs, S., 2016. *How did email grow from messages between academics to a global epidemic?* [Online]
Available at <https://www.theguardian.com/technology/2016/mar/07/email-ray-tomlinson-history> [Accessed 28 August 2020].

Giyanani, R. and Desai, M., 2014. Spam Detection using Natural Language Processing. *IOSR Journal of Computer Engineering* (IOSR-JCE), 16(5), pp. 116-119.

Goundar, S., 2012. Chapter 3 - Research Methodology and Research Method. In: Cloud Computing. Wellington: ResearchGate.
Grayson, M., 2020. *Why is Sender Policy Framework (SPF) not effective in blocking domain spoofing without the implementation of other authentication standards?* [Online]

Available at <https://help.returnpath.com/hc/en-us/articles/223594968-Why-is-Sender-Policy-Framework-SPF-not-effective-in-blocking-domain-spoofing-without-the-implementation-of-other-authentication-standards-> [Accessed 28 September 2020].

Gurin, C. and Fisher, L., 2013. *Email Marketing Benchmarks: Key Data, Trends and metrics.* New York: IBM.

Guzella, T.S. and Caminhas, W.M., 2009. A review of machine learning approaches to Spam filtering. *Expert Systems with Applications*, 36, pp. 206-222.

Halip, M.H.M., Shukran, M.A.M., Zakaria, O. and Zakaria, S.N.A.S., 2012. Detection of steganographic messages in email attachment. *Modern Applied Science,* 6(11), pp. 29-34.

Hamdan, M.A. and Hamarsheh, A., 2017. AH4S: An algorithm of text in text steganography using the structure of omega network. *Security and Communication Networks*, Volume 9, pp. 6004-6016.

Hamid, N., Yahya, A., Ahmad, R. and Al-Qershi, O.M., 2012. Image Steganography Techniques: An Overview. *International Journal of Computer Science and Security (IJCSS),* 6(3), pp. 168-183.

Harris, D., 2003. SpamHelp: Drowning in Sewage: SPAM, the curse of the new millennium: an overview and white paper. [Online]
Available at: https://www.spamhelp.org/articles/Drowning-in-sewage.pdf
[Accessed 15 March 2019].

Hasan, M. F., UzZaman, N. and Khan, M., 2007. Advances and Innovations in Systems, *Computing Sciences and Software Engineering*. Bangladesh, Springer.

Hema, V. and Shyni, E., 2015. DoS Attack Detection Based on Naive Bayes Classifier. *Middle-East Journal of Scientific Research*, vol. 23, pp. 398-405.

Huanhuan, H., Xin, Z., Weiming, Z. and Nenghai, Y., 2017. Adaptive Text Steganography by Exploring Statistical and Linguistical Distortion. *IEEE,* Issue 16, pp. 145-150.

Jabeen, H., 2018. Stemming and Lemmatization in Python. [Online]
Available at: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python [Accessed 10 October 2019].

Jithin, A., 2016. *What is MIME (Multi-Purpose Internet Mail Extension*). [Online]
Available at: https://www.interserver.net/tips/kb/mime-multi-purpose-internet-mail-extensions/#:~:text=The%20MIME%20stands%20for%20Multi,character%20sets%20other%20than%20ASCII. [Accessed 20 September 2020].

Jøsang, A., 2016. *Generalising Bayes' Theorem in Subjective Logic*. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI) held at Baden-Baden on 26th September 2016.

Jurafsky, D. and Martin, H.J., 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed. Colorado: Prentice Hall.

Jurafsky, D. & Martin, J.H., 2019. Chapter 3 - N-gram Language Models. In: Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Stanford: Stanford University, pp. 30-58.

Kadam, S., 2016. *Dependency Parsing in NLP*. [Online]
Available at <https://shirishkadam.com/2016/12/23/dependency-parsing-in-nlp/>
[Accessed 25 September 2019].

Kazdin, A.E., 2003. Methodology: What it is and why it is so important? American Psychological Association, Volume 3rd Edition, p. 5–22.

Khurana, D., Koli, A., Khatter, K. and Singh, S., 2017. Natural Language Processing: State of the Art, Current Trends and Challenges. Accendere Knowledge Management Services.

Kiah, M. L. M., Zaidan, B.B., Zaidan, A. A., Ahmed, A. M. and Al-Bakri, S.H., 2011. A review of audio-based steganography and digital watermarking. *International Journal of the Physical Sciences*, 6(16), pp. 3837-3850.

Klensin, J., 2008. RFC5321: Simple Mail Transfer Protocol. [Online]
Available at: https://tools.ietf.org/html/rfc5321 [Accessed 20 July 2019].

Koluguri, A., Gouse, S. and Reddy, P.B., 2014. Text steganography methods and its tools. *International Journal of Advanced Scientific and Technical Research,* 2(4), pp. 888-900.

Kong, J.S., Rezaei, B.A., Sarshar, N. and Roychowdhury, V.P., 2006. Collaborative Spam Filtering using Em-Mail Networks. *Computer,* 39(8), pp. 68-73.

Kothari, M., 2020. *Feature Extraction Techniques – NLP*. [Online]
Available at <https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/>
[Accessed 8 November 2020].

Kumar, R., 2019. *Word embeddings— Fun with Word2Vec and Game of Thrones*. [Online]
Available at <https://medium.com/@khulasaandh/word-embeddings-fun-with-word2vec-and-game-of-thrones-ea4c24fcf1b8> [Accessed 10 November 2020].

Lai, G.H., Chen, C.M., Liah, C.S. and Chen, T., 2009. A collaborative anti-spam system. *Expert Systems and Applications*, 36(3), pp. 6645-6653.
Leeson, W., Resnick, A., Alexander, D. and Rovers, J., 2019. Natural Language Processing in qualitative public health research: a proof of concept study. *International Journal of Qualitative Methods,* 18, pp. 1-9.

Le, Q. and Mikolov, T., 2014. *Distributed representations of sentences and documents*. Beijing, China, W&CP.

Liddy, E.D., 2001. Natural Language Processing. In Encyclopedia of Library and Information Science. 2nd Edition ed. NY: Marcel Decker, Inc.

Liu, W. and Wang, T., 2011. Online active multi-field learning for efficient email spam. *Knowledge and Information Systems,* 33, pp. 117-136.

Lord, N., 2018. *A definition of email security*. [Online]
Available at <https://digitalguardian.com/blog/what-email-security-data-protection-101>
[Accessed 6 May 2019].

Ma, E., 2018. *Understand how to transfer your paragraph to vector by doc2vec*. [Online]
Available at <https://towardsdatascience.com/understand-how-to-transfer-your-paragraph-to-vector-by-doc2vec-1e225ccf102> [Accessed 23 July 2020].

Marczyk, G., DeMatteo, D. and Festinger, D., 2005. *Essentials of Research Design and Methodology.* Hoboken, New Jersey: Wiley & Sons.

Mason, J., 2006. *SpamAssassin Public Mail Corpus*. [Online]
Available at <https://spamassassin.apache.org/old/publiccorpus/readme.html>
[Accessed 5 July 2020].

McDonald, A., 2004. *SpamAssassin: a practical guide to integration and configuration*. Birmingham, United Kingdom: PACKT Publishing Limited.

McGrath, J., 1982. *Dilemmatics: the study of research choices and dilemmas*. Chicago: Sage, pp. 69-80.

McIntyre, K., Bengfort, B. and Gray, L., 2019. *Yellow Brick - Classification Report*. [Online]
Available at: https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html
[Accessed 26 July 2020].

McKinney, W., 2020. Pandas: Powerful Python data analysis. [Online]
Available at: https://www.learnpython.org/en/Pandas_Basics
[Accessed 10 June 2020].

McLeod, S.A., 2019*. What are independent and dependent variables*. [Online]
Available at: https://www.simplypsychology.org/variables.html [Accessed 4 July 2020].

McNabb, E.D., 2010. *Research methods for political science*. 2nd ed. New York, USA: Routledge Taylor and Francis Group.
Michalsons, 2008. *Guide to the ECT Act in South Africa*. [Online]
Available at: https://www.michalsons.com/blog/guide-to-the-ect-act/81
[Accessed 20 June 2020].

Mikolov, T. and Le, Q.V., 2014. *Distributed Representations of Sentences and Documents*. CoRR, Volume abs/1405.4053.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J., 2013. Distributed Representations of Words and Phrases and their Compositionality. Red Hook, NY, USA, Curran Associates, Inc., pp. 3111 - 3119.

Miyasato, K., 2020. *Classification Report: Precision, Recall, F1-Score, Accuracy*. [Online] Available at <https://medium.com/@kennymiyasato/classification-report-precision-recall-f1-score-accuracy-16a245a437a5> [Accessed 3 August 2020].

Mohammed, S., Mohammed, O., Fiaidhi, J., Fong, S. and Kim, T.H., 2013. Classifying unsolicited. *International Journal of Hybrid Information Technology*, 6(1), pp. 43-55.

Mouton, J., 2004. *How to succeed in your Master's and Doctoral studies*. Pretoria: Van Schaik Publishers.

Muris, T.J., Thompson, M. W., Swindle, O., Leary, T.B. and Harbour, P. J., 2004. *National do not email registry: A report to Congress*, Washington D.C: Federal Trade Commission.

Nechta, I. and Fionov, A., 2011. *Applying statistical methods to text steganography*. Siberian State University of Telecommunication and Information Science, pp. 278-284.

Neubig, G., n.d. *NLP Programming Tutorial 12 - Dependency Parsing*. Ikoma, Nara: NAIST - Nara Institute of Science and Technology.

Nicholson, C., 2019. *A beginner's guide to Word2Vec and Neural Word embeddings*. [Online] Available at <https://pathmind.com/wiki/word2vec#:~:text=Word2vec%20is%20a%20two%2Dlayer,deep%20neural%20networks%20can%20understand > [Accessed 23 July 2020].

Nordquist, R., 2019. *ThoughtCo - Pragmatics gives context to language.* [Online] Available at <https://www.thoughtco.com/pragmatics-language-1691654> [Accessed 30 September 2019].

Nordquist, R., 2020. *Surface structure (generative grammar).* [Online] Available at <https://www.thoughtco.com/surface-structure-transformational-grammar-1692009> [Accessed 5 August 2019].

Indurkhya, N. & Damerau, F., 2010. *Handbook of natural language processing.* Second Edition ed. USA: Chapman & Hall/CRC.

Oates, B.J., 2006. *Researching Information Systems and Computing*. London: Sage Publications.

Okutan, A. and Cebi, P.D.Y., 2019. A framework for cyber-crime Investigation. *Procedia - Computer Science*, 158, pp. 287-294.

Olivier, M.S., 2009. *Information Technology Research.* 3rd ed. Pretoria: Van Schaik Publishers.

Onal, K.D. and Karagoz, P., 2015. *Named entity recognition from scratch on Social Media*. Middle East Technical University, pp. 1-16.

Orman, H., 2015. *Encrypted email - The history and technology of message privacy*. UTAH: Springer.

Patel, M.,2019. *Medium: Naive Bayes - Machine learning algorithm.* [Online] Available at <https://medium.com/@meetpatel12121995/naive-bayes-machine-learning-algorithm-aaf57bdc8d87> [Accessed 26 September 2020].

Patil, T. R. & Sherekar, S. S., 2013. Performance Analysis of Naive Bayes and J 48 Classification Algorithm for Data Classification. International Journal of Computer Science and Applications, 6(2), pp. 256-261.

Peng, T., Harris, I.G. and Sawa, Y., 2018. *Detecting Phishing attacks using Natural Language Processing and Machine Language.* IEEE - International Conference on Semantic Computing, pp. 300-301.

Powers, D.M.W., 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies,* 2(1), pp. 37-63.

Prabhakaren, S., 2018. *Lemmatization approaches with examples in Python*. [Online] Available at <https://www.machinelearningplus.com/nlp/lemmatization-examples-python/> [Accessed 18 September 2019].

Pranjal, P., 2019. *Machine learning: data preprocessing: concepts*. [Online] Available at <https://towardsdatascience.com/data-preprocessing-concepts-fa946d11c825> [Accessed 10 July 2020].

Puertas, E.S., Hidalgo, J.M. G. & Perez, J.C., 2008. Email Spam filtering. *Advances in Computers*, 74, pp. 45-112.

Pupale, R., 2018. *Support Vector Machines (SVM) — an overview*. [Online] Available at <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989#:~:text=SVM%20or%20Support%20Vector%20Machine,separates%20the%20data%20into%20classes > [Accessed 10 November 2020].

Radicatic, 2020. *Email Statistics Report 2020 - 2024*. Palo Alto, USA: The Radicatic Group. Available at <https://www.radicati.com/wp/wp-content/uploads/2020/01/Email_Statistics_Report,_2020-2024_Executive_Summary.pdf>

Rajani, S. and Hanumanthappa, M., 2016. Techniques of semantic analysis for Natural Language Processing - a detailed survey. *International Journal of Advanced Research in Computer and Communication Engineering,* 5(2), pp. 146-149.

Raphael A.J. and Dr Sundaram, V., 2011. Cryptography and steganography - a survey. *International Journal of Computer Technology and Applications*, 2(3), pp. 626-630.

Rehurek, R., 2020. *Gensim: Topic modelling got humans*. [Online]
Available at <https://radimrehurek.com/gensim/> [Accessed 8 November 2020].

Richer, P., 2019. *Steganalysis: detecting hidden information with computer forensics analysis.* (s.l.): SANS Technology Institute - Information Security Reading Room.

Rouse, M., 2005. *TechSecurity - Melissa virus*. [Online]
Available at < https://searchsecurity.techtarget.com/definition/Melissa-virus>
[Accessed 28 September 2018].

Rouse, M., 2010. *TechTarget - UCE* (Unsolicited Commercial Email). [Online]
Available at < https://searchcio.techtarget.com/definition/UCE> [Accessed 20 May 2019].

Roy, S. and Manasmita, M., 2011. A novel approach to format based text steganography. *Association for Computing Machinery (ACM*), pp. 511-516.

Rusland, N.F., Norfaradilla, W., Kasim, S. and Hafit, H., 2017. *Analysis of Naïve Bayes Algorithm for Email Spam.* Malaysia, IOP Publishing.

Sadi, A.G., 2015. Image Steganography Approach. *International Journal of Computer Science and Mobile Computing,* 4(8), pp. 166-169.

Salian, I., 2018. *SuperVize me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* [Online]
Available at <https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/#:~:text=In%20a%20supervised%20learning%20model,and%20patterns%20on%20its%20own > [Accessed 11 November 2020].

Sanz, E.P., Hidalgo, J.M. and Perez, J.P., 2008. Email spam filtering. *Advances in Computers,* 74, pp. 45-107.

Särud, L., 2016. *Misconfigured email servers open the door to spoofed emails from top domains*. [Online]
Available at <https://blog.detectify.com/2016/06/20/misconfigured-email-servers-open-the-door-to-spoofed-emails-from-top-domains/> [Accessed 27 September 2020].

Satterfield, B., 2006. *Ten spam-filtering methods explained.* [Online]
Available at <https://www.techsoupcanada.ca/en/learning_center/10_sfm_explained>
[Accessed 20 July 2018].

Sayad, S., 2010. *Naive Bayesian*. [Online]
Available at < https://www.saedsayad.com/naive_bayesian.htm> [Accessed 30 June 2019].

Schryen, G., 2007. *Anti-Spam - Measures, Analysis and Design.* Aachen, Germany: Springer.

Schwartz, A., 2004. Spam Assassin. Boston, USA: O'Reilly Media, Inc.

Seltzer, L., 2013. *DKIM: useless or just disappointing*. [Online]
Available at < https://www.zdnet.com/article/dkim-useless-or-just-disappointing>
[Accessed 21 September 2020].

Shaikh, R., 2018. *Gentle start to Natural Language Processing using Python*. [Online]
Available at https://towardsdatascience.com/gentle-start-to-natural-language-processing-using-python-6e46c07addf3 [Accessed 8 October 2019].

Sharma, A. Rekha, J. and Manisha, M., 2015. Data Pre-Processing in Spam Detection. International Journal of Science Technology & Engineering, 1(11), pp. 33-37.

Sharma, S., Gupta, A., Trivedi, M.C. and Yadav, V.K., 2016. Analysis of Different Text Steganography Techniques: A Survey. IEEE, pp. 130-133.

Shih, Y.F., 2017. Digital watermarking and steganography: fundamentals and techniques. 2nd ed. New Jersey: CRC Press.

Shrivastava, M., Agrawal, N., Mohapatra, B., Singh, S. and Bhattacharya, P., 2004. *Morphology based Natural Language Processing tools for Indian Languages*. Indian Institute of Technology.

Sinclair, S., 2004. Adapting Bayesian statistical spam filters to the server side. *Journal of Computing Sciences in Colleges,* 16(5), pp. 344-346.

Singh, H., Diwakar, A. and Upadhyaya, S., 2014. A Novel Approach to Text Steganography. International Congress on Computer, Electronics, Electrical, and Communication Engineering, 59(2), pp. 7-12.

Singh, H., Singh, P. K. and Saroha, K., 2009. A Survey on Text Based Steganography. Noida, India, Proceedings of the 3rd National Conference.

Singh, N.G., 2017. *Overview of Artificial Intelligence and Natural Language Processing*. [Online]
Available at <https://www.upwork.com/hiring/for-clients/artificial-intelligence-and-natural-language-processing-in-big-data/ > [Accessed 25 September 2019].

Singh, S., 2018. *Natural Language Processing for Information Extraction. arxiv.org*, pp. 1-16.

Sochor, T., 2014. *Overview of e-mail SPAM elimination and its efficiency*. Marrakech, Morocco, Institute for Electronic and Electrical Engineers (IEEE).

StanfordNLP Group, 2018. *The Stanford Natural Language Processing - Coreference Resolution.* [Online]
Available at <https://nlp.stanford.edu/projects/coref.shtml> [Accessed 15 October 2019].

Stecanella, B., 2019. *A practical explanation of the Naive Bayes' Classifier*. [Online]
Available at < https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/>
[Accessed 28 June 2019].

Stehman, S.V., 1997. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1), pp. 77-89.

Steven, L., 2020. *TextBlob: Simplified Text Processing*. [Online]
Available at: https://textblob.readthedocs.io/en/dev/ [Accessed 8 November 2020].

Stine, K. and Scholl, M., 2010. E-mail Security: an overview of threats and safeguards. *Journal of AHIMA 81,* vol. 4, pp. 28-30.

Su, X. and Khoshgoftaar, T.M., 2009. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, vol. 2009, pp. 1-14.

Tamir, D., 2013. *FBI warns of increase in spear-phishing attacks*. [Online]
Available at <https://securityintelligence.com/fbi-warns-increase-spear-phishing-attacks/>
[Accessed 9 August 2017].

Taouil, Y., Ameur, E.B., Souhar, A., Harraj, A.El. and Belghiti, M.T., 2016. High imperceptibility image Steganography Methods based on HAAR DWT. *International Journal of Computer Applications*, 138(10), pp. 38-43.

Taskiran, C.M., Topkara, U., Topkara, M. and Delp, E.J., 2006. Attacks on lexical natural language steganography systems. *Proc. SPIE*, Vol. 6072, pp. 607209-1 607209-9.

Tazzyman, S., 2020. *NLP Guidance - Feature Extraction*. [Online]
Available at < https://moj-analytical-services.github.io/NLP-guidance/FeatureSelection.html#:~:text=We%20want%20to%20embed%20our,and%20what%20can%20be%20ignored> [Accessed 22 July 2020].

Teja, S., 2020. *Stop Words in NLP*. [Online]
Available at <https://medium.com/@saitejaponugoti/stop-words-in-nlp-5b248dadad47>
[Accessed 2 November 2020].

Trovi, D.S., ShivarKumar, K.B. and Das, R., 2016. A unique data security using text steganography. *IEEE*, pp. 3834 - 3838.
Turian, J., Ratinov, L. and Bengio, Y., 2010. *Word representations: a simple and general method for semi-supervised learning.* Uppsala, Sweden, Association for Computational Linguistics.

Christina, V., Karpagavalli, S. and Suganya, G., 2010. A Study on Email Spam Filtering Techniques. International Journal of Computer Applications, 12(1), pp. 7-9.

Vladimir, F., 2018. 5 *Heroic Tools for Natural Language Processing*. [Online]
Available at <https://towardsdatascience.com/5-heroic-tools-for-natural-language-processing-7f3c1f8fc9f0> [Accessed 16 September 2019].

Vyas, T., Prajapati, P. and Gadhwal, S., 2015. A survey and evaluation of Supervised Machine Learning Techniques for Spam E-Mail Filtering. Ahmedabad, Institute for Electronic and Electrical Engineers (IEEE).

Wang, H. and Wang, S., 2004. Cyber Warfare: Steganography vs. Steganalysis. *Communications of the ACM*, 47(10), pp. 76-82.

Weiss, E., 2018. *Understanding SPF and DKIM to improve email deliverability*. [Online]
Available at: https://www.sparkpost.com/blog/understanding-spf-and-dkim
[Accessed 30 September 2020].

Wenxuan, S. and Maoqiang, X., 2013. A reputation-based collaborative approach for spam filtering. *AASRI Procedia*, Vol. 5, pp. 220-227.

Williams, C., 2007. Research methods. *Journal of Business & Economic Research*, 5(3), pp. 65-72.

Wozniak, P., 2004. *E-mail, incremental reading, creativity, and time-management*. [Online]
Available at <https://www.supermemo.com/articles/e-mail.htm> [Accessed 16 September 2018].

Xiang, L., Yu, J., Yang, C., Zeng, D. and Shen, X., 2018. A word-embedding-based steganalysis method for linguistic steganography via synonym substitution. *IEEE*, Vol. 6, pp. 64131 - 64141.

Young, N., 2019. *What is ethnography research?* [Online]
Available at <https://www.experienceux.co.uk/faqs/what-is-ethnography-research/>
[Accessed 13 June 2020].

Yu, S., 2015. Covert communication by means of email spam: a challenge for digital investigation. *Digital Investigation*, vol. 13, pp. 72-79.

Zhi-li, C., Liu-Sheng, H., Zehn-Shan, Y., Ling-Jun, L. and Wei, Y., 2008. A statistical algorithm for Linguistic Steganography Detection based on Distribution of Words. *IEEE - Computer Society*, pp. 558-563.

Zhou, W., Wen, Junhao., Qu, Qiang., Jun, Zeng. and Tian, C., 2018. Shilling attack detection for recommender systems based on creditability of group users and rating time series. *PloS ONE*, 13(5), pp. 1-17.

Addendum

The source code and corpora used in this research can be downloaded from the following link:

Source code and corpora:
https://1drv.ms/u/s!Ag5_OYUcYNFMfMNiM_SxQDlm7ps?e=yccZan

The prototype was developed using the Spyder Integrated Development Environment running at least Python version 3.5. Spyder can be downloaded from the following link:

Spyder: https://www.spyder-ide.org/