APPLYING INSIGHTS FROM MACHINE LEARNING TOWARDS GUIDELINES FOR THE DETECTION OF TEXT-BASED FAKE NEWS

OKUHLE NGADA

2021

APPLYING INSIGHTS FROM MACHINE LEARNING TOWARDS GUIDELINES FOR THE DETECTION OF TEXT-BASED FAKE NEWS

By

Okuhle Ngada

Submitted in fulfilment of the requirements for the degree of

Master of Information Technology

to be awarded at the

Nelson Mandela University

December 2021

Supervisor: Prof. Bertram Haskins

NELSON MANDELA
UNIVERSITY

PERMISSION TO SUBMIT FINAL COPIES OF TREATISE/DISSERTATION/THESIS TO THE EXAMINATION OFFICE

Please type or complete in black ink

FACULTY: Faculty of Engineering, the Built Environment and Technology

SCHOOL/DEPARTMENT: School of Information Technology

I, (surname and initials of supervisor) Haskins, BP

and (surname and initials of co-supervisor) ____________________

the supervisor and co-supervisor respectively for (surname and initials of candidate) Ngada, O

(student number) 213215136 a candidate for the (full description of qualification)

Master of Information Technology

with a treatise/dissertation/thesis entitled (full title of treatise/dissertation/thesis):

Applying Insights From Machine Learning Towards Guidelines for the Detection of Text-Based Fake News

It is hereby certified that the proposed amendments to the treatise/dissertation/thesis have been effected and that **permission is granted to the candidate to submit** the final copies of his/her treatise/dissertation/thesis to the examination office.

____________________	1 November 2021
SUPERVISOR	**DATE**

And

____________________	____________________
CO-SUPERVISOR	**DATE**

DECLARATION

I, *Okuhle Ngada s213215136*, hereby declare that the treatise/ dissertation/ thesis for Students qualification to be awarded is my own work and that it has not previously been submitted for assessment or completion of any postgraduate qualification to another University or for another qualification.

O. Ngada

____

Okuhle Ngada

Official use:

In accordance with Rule G5.11.4,

5.11.4 A treatise/dissertation/thesis must be accompanied by a written declaration on the part of the candidate to the effect that it is his/her own work and that it has not previously been submitted for assessment to another University or for another qualification. However, material from publications by the candidate may be embodied in a treatise/dissertation/thesis

Abstract

Web-based technologies have fostered an online environment where information can be disseminated in a fast and cost-effective manner whilst targeting large and diverse audiences. Unfortunately, the rise and evolution of web-based technologies have also created an environment where false information, commonly referred to as “fake news”, spreads rapidly. The effects of this spread can be catastrophic.

Finding solutions to the problem of fake news is complicated for a myriad of reasons, such as: what is defined as fake news, the lack of quality datasets available to researchers, the topics covered in such data, and the fact that datasets exist in a variety of languages. The effects of false information dissemination can result in reputational damage, financial damage to affected brands, and ultimately, misinformed online news readers who can make misinformed decisions.

The objective of the study is to propose a set of guidelines that can be used by other system developers to implement misinformation detection tools and systems. The guidelines are constructed using findings from the experimentation phase of the project and information uncovered in the literature review conducted as part of the study. A selection of machine and deep learning approaches are examined to test the applicability of cues that could separate fake online articles from real online news articles. Key performance metrics such as precision, recall, accuracy, F1-score, and ROC are used to measure the performance of the selected machine learning and deep learning models. To demonstrate the practicality of the guidelines and allow for reproducibility of the research, each guideline provides background information relating to the identified problem, a solution to the problem through pseudocode, code excerpts using the Python programming language, and points of consideration that may assist with the implementation.

Acknowledgements

I would like to thank my supervisor, Prof. Bertram Haskins for the expertise, patience, motivation, and support received throughout this study. This guidance has greatly contributed to the successes contained in this project. I also take the opportunity to thank the following entities for their support during this study:

- The financial assistance I received from the Nelson Mandela University, through the Postgraduate Research Scholarship has been greatly acknowledged and appreciated.
- The proofreading and editing service offered by Barrie Lake has been greatly appreciated.

I would also like to thank my mother, Bulelwa Ngada, my late father, Leonard Lumkile Ngada, my siblings, Tuleka Ngxabani, Unathi Ngada, and Sakhile Ngada, and my family for the support, inspiration and encouragement received during this study. Finally, I would also like to thank my friends, who supported me throughout the study.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BERT: Bidirectional Encoder Representations from Transformers 54

BiMPM: Bi-Directional Multi-Perspective Matching for Natual Language Sentences 29

CFS: Checkworthy Factual Sentence 42

CNN: Convolutional Neural Network 49

CTTR: Corrected Type Token Ratio 99

cuDNN: Nvidia CUDA Deep Neural Network 87

FN: False Negative 44

FP: False Positive 44

GloVe: Global Vectors for Word Representation 94

GPT: Generative Pre-Trained Transformer 39

GRU: Gated Recurrent Unit 48

HD-D: Hypergeometric Distribution Diversity 99

HFLU: Hybrid Feature Learning Unit 52

LSTM: Long Short Term Memory 48

MATTR: Moving Average Type-Token Ratio 99

MSTTR: Mean Segmental Type-Token Ratio 99

MTLD: Measure of Lexical Diversity 99

NBC: Naive Bayes Classifier 42

NFS: Non Factual Sentence 42

NLP: Natural Language Processing 1

NLTK: Natural Language Toolkit 98

OOV: Out of Vocabulary 86

POS: Parts of Speech 35

PV-DBOW: Paragraph Vector - Distributed Bag of Words 68

PV-DM: Paragraph Vector - Distributed Memory 68

ReLU: Rectified Linear Unit 47

RF: Random Forest 43

RNN: Recurrent Neural Network 47

RTTR: Root Type-Token Ratio 99

T5: Text-To-Text Transfer Transformer 68

TI-CNN: Text and Image Convolutional Neural Network 95

TN: True Negative 44

TP: True Positive... 44

TTR: Type-Token Ratio.. 99

UFS: Uninportant Factual Sentence.. 42

1 Chapter 1 - Introduction

The world has entered an era where facts are being replaced by emotions and personal beliefs. The truth is not easy to identify at first glance, given the rise of online web technologies facilitating the easy spread of information. Fake News is a catchphrase used to discredit sources lacking substantial evidence (Fernandez, 2017). The catchphrase has seen a rise in popularity thanks to social media, mainstream news firms, and public figures. The term has a much broader meaning – widely accepted definitions are clickbait, propaganda, commentary/opinion, and humour/satire (Campan, Cuzzocrea, & Truta, 2017). Social media platforms, news firms, individuals and organisations with malicious intent have aided in the propagation of fake news.

1.1.1 The Global Fake News Epidemic

Social networking platforms have demonstrated the wide reach and impact they have on society. Facebook has a global user base of 2.2 billion, while Twitter has 310 million users worldwide (Statistica, 2018). In the South African social networking space, Goldstuck & Patricios (2017) note 16 million Facebook users and 8 million Twitter users. Social media use is widespread, and many posts shared on these platforms relate to the dissemination of news. In a report published by Stockling, Barthel, & Grieco (2018), which sampled 9.7 million tweets on the topic of immigration, it was found that 75% of the tweets had a link to at least one online news organisation. In the subset of tweets that included links to any websites, 42% of such tweets linked to news organisations, while 29% linked to other information providers, such as blogs, non-profit websites and governmental websites (Stockling, Barthel, & Grieco, 2018).

At the same time, advancements in technology make it faster, simpler, and cheaper to create professional websites. Such advancements have been exploited to create polished websites, capable of generating substantial revenue (Fernandez, 2017). Online fake news websites often generate revenue by displaying adverts on related websites, through online advertising networks. During the 2016 US Elections, teenagers in the city of Veils, Macedonia created 100 websites that favoured presidential candidate, Donald Trump (Subramanian & Martin, 2017). One of the teenagers reported he had earned as much as $16,000 from running online advertisements through advertising networks (Subramanian & Martin, 2017). Allcott &

Gentzkov (2017) state two reasons which motivate an author to create false content. The first reason is the substantial monetary gains made through running adverts for advertising networks. The second reason is to spread the ideals of the author (Allcott & Gentzkov, 2017).

1.1.2 Fake News in South Africa

In recent years, South Africa has experienced several instances of misinformation dissemination. The effects of such misinformation in South Africa can be seen through high racial tensions, a general distrust in the government and related organisations, and the shifting viewpoints of many South Africans. The most notable instances of false information dissemination in the country include the Bell Pottinger scandal, and the former UNISA employee exposed for operating a network of fake news websites.

During a time where state capture took the centre stage of South African politics, Wasserman (2017) found that the infamous Gupta family used PR firm, Bell Pottinger, to create a storyline that the supporters of the former president, Jacob Zuma, were victims of ‘white monopoly capital’. Media houses and organisations, such as Africa News Network 7, The New Age, and Black First Land First (BLF) were linked to tweets aimed at countering allegations made in the former Public Protector’s report on state capture (Wasserman, 2017).

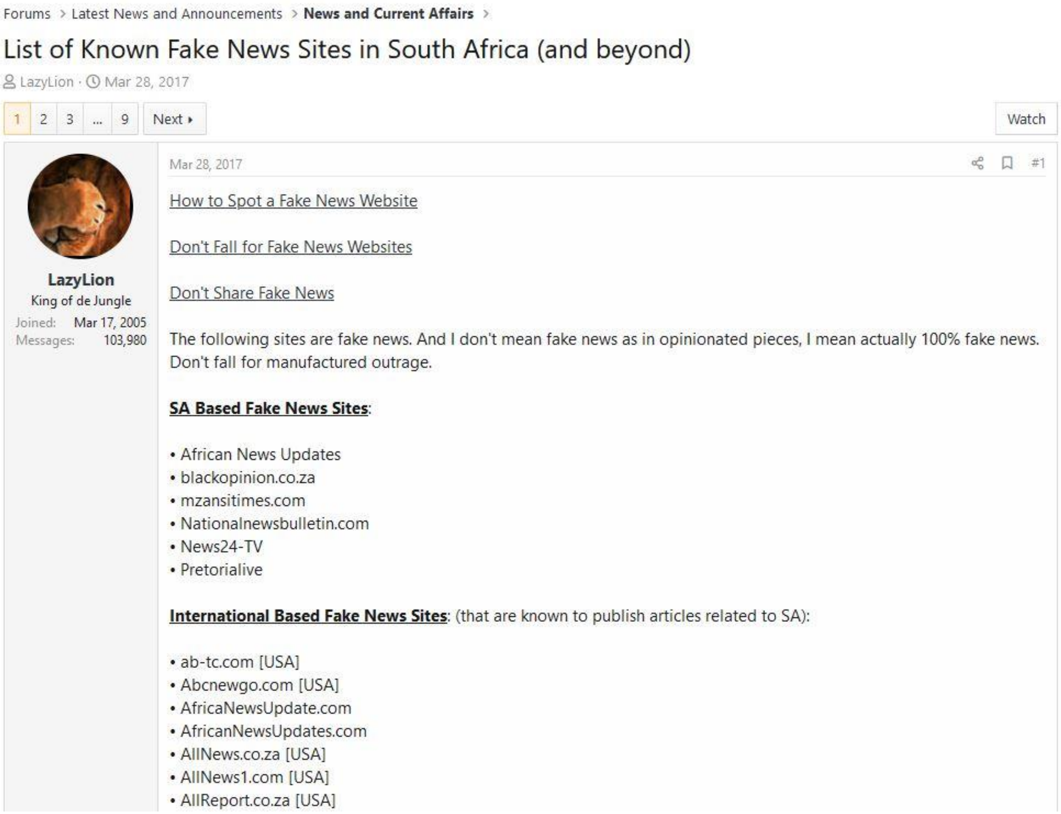
In an investigation led by popular South African news firm News24, it was found a University of South Africa (UNISA) employee and his sibling were responsible for a series of fake news websites and social media accounts (le Roux, 2018). The authors generated advertising revenue through 15 of the fake news websites, by using a single Google AdSense ID (le Roux, 2018).

In the cases briefly introduced, it is evident that reputable organisations, such as well-known news firms and agencies, can be used to spread false narratives on topics. The Bell Pottinger scandal and the related fake tweets have played a role in rising racial tensions and mistrust in South Africa. Online news consumers are led to formulate misguided opinions on organisations, brands, or people due to consuming fake news. The rapid spread of information online makes the task of manually identifying fake and real news articles a time-consuming, and complicated task.

1.1.3 Current Initiatives in Addressing the Spread of Fake News

The rapid advancements in technology have facilitated the formation of many online fact-checking organisations. A few of the most popular organisations are Snopes (https://www.snopes.com/), Africa Check (https://africacheck.org/), PolitiFact (https://www.politifact.com/) and FactCheck (https://www.factcheck.org/). Fact check organisations employ teams of journalists who verify claims and the authenticity of articles on the internet (Batchelor, 2017). Though many fact-checking organisations exist, Batchelor (2017) notes these organisations vary in thoroughness, and points out such organisations have shown bias (Batchelor, 2017).

Other initiatives aimed at combatting false information dissemination include the contributions made by Zimdar (2016). Zimdar (2016) provides a collection of fake news websites, and a set of guidelines that assist with assessing the credibility of news sources (Zimdar, 2016). In South Africa, forum users of the popular website, MyBroadband, created a forum thread where a collection of fake news sites are regularly updated (MyBroadband, 2017).



Forums > Latest News and Announcements > **News and Current Affairs** >

List of Known Fake News Sites in South Africa (and beyond)

LazyLion · Mar 28, 2017

How to Spot a Fake News Website

Don't Fall for Fake News Websites

Don't Share Fake News

The following sites are fake news. And I don't mean fake news as in opinionated pieces, I mean actually 100% fake news. Don't fall for manufactured outrage.

SA Based Fake News Sites:

- African News Updates
- blackopinion.co.za
- mzansitimes.com
- Nationalnewsbulletin.com
- News24-TV
- Pretorialive

International Based Fake News Sites: (that are known to publish articles related to SA):

- ab-tc.com [USA]
- Abcnewgo.com [USA]
- AfricaNewsUpdate.com
- AfricanNewsUpdates.com
- AllNews.co.za [USA]
- AllNews1.com [USA]
- AllReport.co.za [USA]

Figure 1.1: List of known fake news websites curated by forum users of the MyBroadband forums (MyBroadband, 2017).

1.1.4 Machine Learning and Fake News Detection

Research in the Natural Language Processing (NLP) and Machine Learning fields have shown the usefulness and effectiveness of such technologies in the detection of fake news. Advancements in the availability of data (in terms of dataset size, lengths of texts, and languages used) have facilitated research capable of producing state-of-the-art results. A summary of such developments is described below.

Vlachos and Riedel (2014) released a dataset consisting of statements taken from Channel 4 and PolitiFact. Both websites maintain collections of statements, which have been verified (Vlachos & Riedel, 2014a). Noting the shortcomings in the dataset released by Vlachos & Riedel (2014), Wang (2017) released the LIAR dataset, consisting of 12 836 statements. Dyson & Golab (2017) used datasets authored by OpenSources (n.d.) and Signal Media (2016) to perform the fake news detection classification task, using 5 machine learning algorithms, namely Support Vector Machines, Stochastic Gradient Descent, Gradient Boosting, Bounded Decision Trees and Random Forests. Hassan, Arslan, Li, & Tremayne (2017) created ClaimBuster – a system that uses a selection of machine learning algorithms, such as Multinomial Naïve Bayes Classifier, Support Vector Machine and the Random Forest to classify statements into one of three possible classes. Ahmed, Traore, & Saad (2017) perform the task of fake news detection, using 6 machine learning algorithms, namely, Support Vector Machine (SVM), Linear Support Vector Machine (LVSM), K-Nearest Neighbour (KNN), Decision Tree (DT), Stochastic Gradient Descent (SGD) and Linear Regression (LR) and a custom dataset that sources real online news articles from Reuters, an online news firm, and a fake online news collection from a dataset hosted on Kaggle.com (Ahmed et al., 2017).

From a text-analysis perspective, researchers have highlighted some common differences that should be considered when discerning false from real news articles. Conroy, Rubin, & Chen (2015) mention two key differences that can be used in differentiating false articles from real articles. The first difference is observed by assessing the correlation between an article's title and body – article titles and contents in fake news articles tend not to correlate (Conroy, Rubin, & Chen, 2015). The second difference, according to Conroy, Rubin, & Chen (2015), can be observed in articles that use questioning headlines when the contents of the article may not be truthful. The authors add that questioning headlines make online news readers assume the article is truthful (Conroy, Rubin, & Chen, 2015). Such differences can be quantified to a numeric value. Wu, Cheng, & Chai (2017) employ cosine similarity to calculate the similarity of an article's headline to the article body. Choudary & Arora (2020) use parts of speech tags to quantify grammar usage in articles. When it comes to online reviews, Feng & Hirst (2013) state two attributes found in fake online reviews. Fake reviews contain contradictions, and they often fail to mention aspects seen in other truthful reviews (Feng & Hirst, 2013).

1.2 Research Problem

Fake news-based misinformation has many consequences such as distrust in firms, brands, or organisations, as well as social and economic ramifications. Online news readers cannot accurately differentiate fake news articles from real news articles due to the ease of creating fake news websites that appear professional and legitimate. This problem is further exacerbated by inadequate fact verification on social media websites. Additionally, the rapid spread of information online makes manual fact verification a time-consuming task. Though numerous fact-checking organisations exist, such organisations have had their thoroughness questioned, and in some cases, had been criticised for showing bias.

1.3 Problem Statement

Due to the rapid spread of online misinformation, in the form of fake news, manual approaches employed to differentiate fake news from real news articles, are time consuming and inadequate, given the rapid dissemination of online news.

1.4 Thesis Statement

Using common indicators that differentiate fake online news articles from real online news articles, machine learning approaches could be used to automate the detection of fake news articles.

1.5 Research Objectives

The defined primary research objective for the study is:

Develop guidelines for the use of machine learning to identify text-based fake news.

To support the primary research objective, three secondary research objectives are defined:

1. *Identify the motives behind the propagation of fake news.*
2. *Identify appropriate indicators, from literature, which could suggest an article is fake news.*
3. *Validate the identified indicators through the application of selected machine learning algorithms.*

1.6 Research Methodology

The project employs numerous quantitative research methods, following a systematic approach. Research methodology describes a systematic way of undertaking research and solving research questions (Sarkar & Sahu, 2018). Quantitative research methods collect numeric, statistical, measurable data to provide answers for a phenomenon (Earl, 2010). Figure 1.2 illustrates the research process undertaken for this project. This study employs the following research methods: Literature Review, Experimentation, and Argumentation.

1.6.1 Literature Review

The objective of the literature review is to paint a clear picture as to what fake news entails, uncover what academics in the fake news research realm have published, as well as serve as guidance in this research project. The literature review identifies gaps in current research and allows the researcher to place their work relative to current findings (Jaidka, Khoo, & Na, 2013). Research in automated fake news detection is particularly useful in this project – it gives insight into what techniques are most applicable, as well as highlighting shortfalls of current automated fake news implementations. The literature review defines possible indicators of articles being fake news, and it makes a case as to why fake news is a problem worth investigating. The literature review addresses secondary research objectives 1 and 2, making clear the common characteristics and criteria constituting fake news, as well as the motive behind the propagation of fake news. To better present the case of why fake news is a problem, and what parties stand to gain or lose from the spread of fake news, it is necessary to unpack the reasons why an individual or organisation would create fake news.

1.6.2 Experimentation

Experiments are performed to test a theory, prove a theory, or explore an idea (Olivier, 2009). The purpose of using experimentation in this project is to validate the effectiveness of indicators, highlighted in the literature review, using machine learning. Ahmed, Traore, & Saad (2017) explore the applicability of n-gram features for fake news detection, using 6 machine learning algorithms namely, Stochastic Gradient Descent (SGD), Support Vector Machine (SVM), Linear Support Vector Machine (LSVM), K-Nearest Neighbour (KNN) and Decision Tree (DT) (Ahmed, Traore, & Saad, 2017). For deep learning, Yang, et al. (2018) used a convolutional neural network architecture as part of a model for fake news detection. Experimentation addresses the secondary research objective 3.

1.6.3 Argumentation

Argumentation aims at convincing critics of the acceptability of your viewpoint, using a collection of statements (usually facts) as the premise (Van Eemeren & Grootendorst, 2004). Argumentation is used in defining an initial set of fake news identifiers from the literature in conjunction with selecting applicable machine learning and deep learning algorithms to test the effectiveness of the identified fake news indicators. The premise of arguments is based on findings uncovered in the literature review, in conjunction with findings from the experimentation phase of the project. Argumentation addresses secondary objectives 2 and 3.

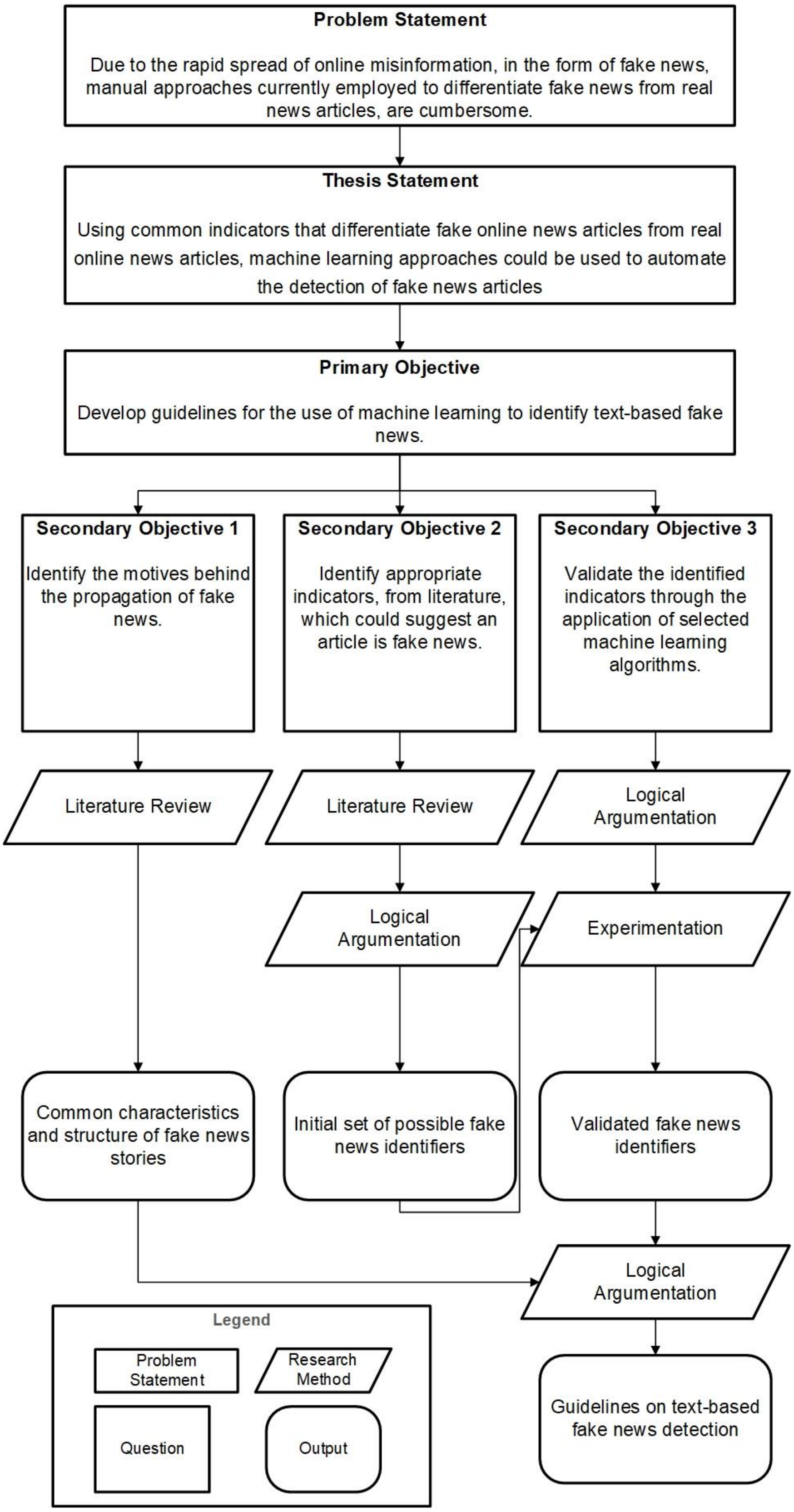

Problem Statement
Due to the rapid spread of online misinformation, in the form of fake news, manual approaches currently employed to differentiate fake news from real news articles, are cumbersome.
Thesis Statement
Using common indicators that differentiate fake online news articles from real online news articles, machine learning approaches could be used to automate the detection of fake news articles
Primary Objective
Develop guidelines for the use of machine learning to identify text-based fake news.
Secondary Objective 1
Identify the motives behind the propagation of fake news.
Secondary Objective 2
Identify appropriate indicators, from literature, which could suggest an article is fake news.
Secondary Objective 3
Validate the identified indicators through the application of selected machine learning algorithms.
Literature Review
Literature Review
Logical Argumentation
Logical Argumentation
Experimentation
Common characteristics and structure of fake news stories
Initial set of possible fake news identifiers
Validated fake news identifiers
Logical Argumentation
Guidelines on text-based fake news detection
Legend
Problem Statement
Research Method
Question
Output

Figure 1.2: Research process diagram for this research project

1.7 Delineation

Publicly available datasets of online news articles, authored by Bisaillon (2020) and Szpakowski (2018), are employed as sources of data for the experimentation phase of the project. For experimentation purposes, the task of fake news detection is treated as a binary classification problem. The experimentation phase of the project is conducted using only historic datasets; it is not applied to a live, real-world environment.

1.8 Ethical Consideration

In the experimentation phase, the required data is sourced from multiple, full online news datasets, which are curated and maintained by various organisations for fake news research. Furthermore, the related organisations have licensed such datasets free for the public to use. University ethical clearance, in connection to this project, is not required as this project's primary source of data relies on publicly available fake news datasets.

1.9 Layout of the Thesis

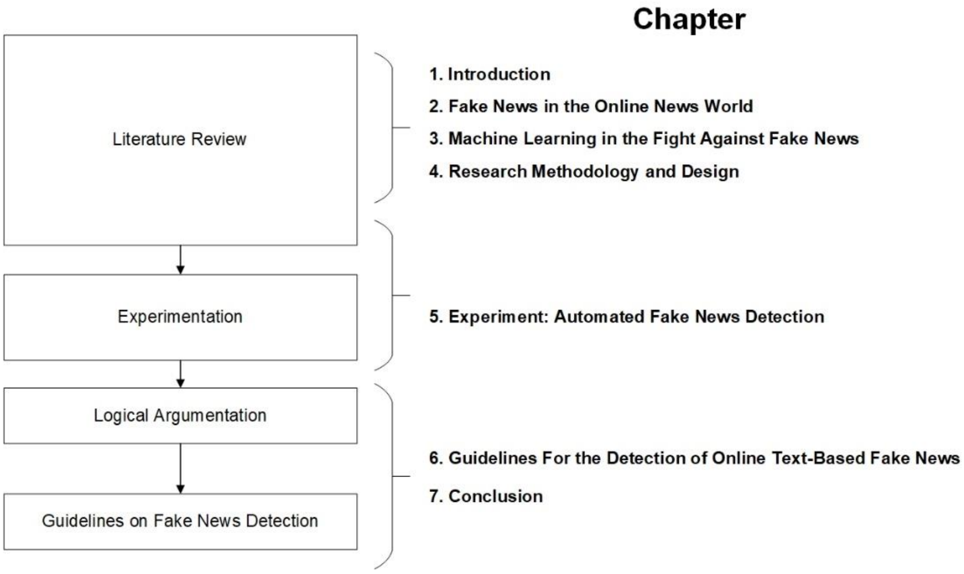

Chapter
Literature Review
Experimentation
Logical Argumentation
Guidelines on Fake News Detection
1. Introduction
2. Fake News in the Online News World
3. Machine Learning in the Fight Against Fake News
4. Research Methodology and Design
5. Experiment: Automated Fake News Detection
6. Guidelines For the Detection of Online Text-Based Fake News
7. Conclusion

Figure 1.3: Chapter layout for the research project

Chapter 1 introduces the domain area of this research project. The chapter provides a high-level view of fake news and the problems the info-demic presents. The effects of such misinformation, in a global and South African context, are also presented. The chapter provides an overview of some related work carried out by authors who have investigated the use of machine learning in the detection of fake news. These works serve as the necessary foundation needed to prove automated fake news detection is possible. Using the high-level view, defined in Chapter 1, **Chapter 2** further expands on this by reporting on fake news in a computational context. This chapter describes what fake news is, explains why it is a problem that should be resolved, and highlights current difficulties in addressing this phenomenon. The chapter ends off by introducing various solutions tested by leading researchers in this realm. Using the suggested solutions from Chapter 2, **Chapter 3** reports on machine learning and deep learning approaches and how these methods have been used to tackle online misinformation. Extensive coverage of related works is also provided. **Chapter 4** expands on the research methodology and design undertaken in this research project. Using background information uncovered in Chapters 2 and 3. **Chapter 5** deals with the experimentation component of the project. Using possible indicators for fake news, as highlighted in Chapter 2, and the available machine learning and deep learning approaches mentioned in Chapter 3, the effectiveness of such indicators and the viability of automated fake news detection is thoroughly examined. Chapter 5 closes with a discussion and further analysis of the results obtained, and how the experimentation observations and results compare with existing literature. **Chapter 6** presents a set of guidelines for the detection of text-based fake news stories. **Chapter 7** summarizes the work undertaken in this project and concludes the research project.

1.10 Conclusion

This chapter introduces fake news and why it is a problem worthy of investigation. Machine learning, and its potential use for mitigating the fake news problem, is also introduced. To solve the issue highlighted in the problem area, a primary objective and a series of secondary objectives are stated. Finally, the research process, delineation, and chapter layout are presented. Chapter 2 provides more background on the concept of fake news by exploring literature on the problem, discussing how fake news benefits its authors while negatively impacting the public, and examining current solutions to the problem.

2 Chapter 2 - Fake News in the Online News World

2.1 Introduction

Advancements in technology have made the global communication and distribution of information easier, more effective, and more cost-efficient. These advancements have given rise to social networking platforms like Facebook and Twitter, which are implicated in the spread of online misinformation. News firms use these networks to expand their reach and rapidly engage with their audience. Bloggers, public figures, and small organisations enjoy the same, cost-effective power, vested in technology when it comes to communicating and distributing information. At the same time, several issues arise, including the credibility of information distributed and the potential dangers surrounding the distribution of false information. The dissemination of fake news is a multi-dimensional problem, which this chapter aims at unpacking.

Rochlin (2017) describes fake news as falsified stories which are published on websites built to imitate the aesthetics and functionality of legitimate news websites. Rubin, Chen, & Conroy (2015) expand on 'fake news', in that it presents itself in many forms, namely, satire, bad reporting, false stories, rumours, and hoaxes, and note the rapid, widespread dissemination of such information on the internet. Kshetri & Voas (2017) state fake news has two general classes, namely disinformation and misinformation – disinformation is false information distributed with the purposeful intent to mislead its audience, while misinformation is incorrect information shared without necessarily malicious intent. Large media organisations realise their influence on society, and as such, these organisations have been known to manipulate information (Granik & Mesyura, 2017). Granik & Mesyura (2017) suggest the objective of fake news is to influence a reader's opinion on various matters. The effects of fake news can be seen through confusion among online readers, distrust in large firms, organisations, and entities, as well as negative economic effects on respective organisations. With the proliferation of fake news, news consumers may find it harder to differentiate real news from fake news and may increase their scepticism of real news (Kogan, Moskowitz, & Niessner, 2019).

Fake news is not limited to text; false information can be disseminated in the form of manipulated images and video. YouTube, a popular video platform, allows users to upload videos and monetize them (Vishwakarma, Varshney, & Yadav, 2019). The headline/body dissonance, as described by Conroy, Rubin, & Chen (2015) applies to some YouTube videos, in that some users upload videos where the title of the video does not coincide with the video's content, in an attempt to sway uses towards their channel (Vishwakarma, Varshney, & Yadav, 2019). Computer algorithms derived from fake news research exist in two categories: image-based algorithms and text-based algorithms. The authors state that the majority of researchers work with text-based features, thus resulting in limited research on image-based fake news detection (Vishwakarma, Varshney, & Yadav, 2019).

2.2 Social Media and Online Misinformation

Researchers in the fake news community have made several findings on the influence social networking platforms have on the dissemination of online fake news. A 2017 online publication by the Pew Research Center reveals key online news readers' behaviour. Gottfried & Shearer (2017) report 67% of Americans use social networks as a source of online news. The figure increased by 5% compared to 2016, where it was reported 62% of Americans used social media as a source of online news (Gottfried & Shearer, 2017). In a 2018 online publication by the Pew Research Center, it was found that 57% of surveyed adults believe that information served on social media is highly inaccurate (Matsa & Shearer, 2018). Similarly, in a survey conducted by Singhal, Shah, Chakraborty, Kumaraguru, & Satoh (2019), it was found that 44,3% of users could not differentiate fake news articles from real news articles. The survey forms part of the manifestation of the SpotFake framework, proposed by Singhal et al., (2019).

Social networking platforms are aware of the detrimental effects relating to fake news and are working to address the problem. Facebook's efforts revolve around three key areas: eliminate the economic benefits associated with the spread of fake news, build products that will limit the spread of false information, and aid people in making better decisions when encountering fake news (Mosseri, 2017). Part of the company's strategy includes working with third-party fact-checking organisations, enforcing stricter advertisement-related policies for individuals known for posting fake news, and employing machine learning technology in detecting fraudulent and fake Facebook accounts (Mosseri, 2017).

Google collaborates with various organisations in addressing the spread of misinformation. The technology-based organization co-launched First Draft, a non-profit organisation, aimed at addressing the spread of misinformation and disinformation through technology and collaborations with other institutes (Google, 2018)

The rise of social media being the primary source of information for many online news readers reaffirms the view of social media platforms contributing to the dissemination of false information.

2.3 What Makes Fake News a Problem

To further understand fake news, and how it has plagued modern society, this section aims at unpacking what makes fake news a 'problem'. Hyman (2018) describes how informing people to simply do their research on a topic is inadequate; most individuals often lack time, motivation, and resources to perform such a task. Kshetri & Voas (2017) note some consumers (online news readers) find it difficult to assess the quality and credibility of information. In a survey conducted by Common Sense Media, among 853 children and teenagers between the ages of 10 – 18, 44% of participants claimed they could tell a fake news story from a real one, and 31% shared a story online, only to find out a few months later that the story was fake (Robb, 2017). Hassan, Arslan, Li, & Tremayne (2017) add that fact-checkers cannot keep up with the rate at which information is spread due to this task being constrained by time, labour, and human intellect. It is near impossible to manually label fake news and real news, given its rapid spread on social media and the immense amount of content published (Shu, Mahudeswaran, & Liu, 2018). Aiding the spread of fake news on social media platforms is the use of bots, which can post information on social media and participate in engage in community discussions (Zhang & Ghorbani, 2019). Such bots are created by humans with the intent of disseminating false information. From a legal perspective, several countries have the right to free speech as part of their constitution. As such, law enforcement agencies cannot easily monitor and censor free speech, which fake news uses (Kshetri & Voas, 2017). One of the most notable cases where fake news had a major influence was in the 2016 US elections (Hassan et al., 2017). This is expanded in section 2.5 of this chapter.

2.4 The Economics Behind the Fake News Epidemic

Kshetri & Voas (2017) note how advertising networks care less about the websites running their advertisements. The website should adhere to guidelines stipulated by the advertising network, and website traffic should be legitimate, and not be generated through bots (Kshetri & Voas, 2017).

Boris, a teenager from a small town called Veils, in Macedonia, revealed how he made $16 000 US dollars in advertising revenue from just two of the 100 pro-Trump fake news websites he created (Subramanian & Martin, 2017). In 2016, National Public Radio (NPR) tracked down Justin Coler, the owner of Disinfomedia – a company that owns many fake news websites. Coler revealed he had approximately 20 to 25 writers, and his fake news websites at the time made their revenue through online ads (Sydell, 2016). Coler suggests other fake news owners make $10 000 to $30 000 US dollars a month (Sydell, 2016).

Kshetri & Voas (2017) suggest 3 reasons that may drive an individual or organization to the creation of fake news. Monetary gain, the psychological impact, and the reduced legal risk are some factors considered by authors of fake news (Kshetri & Voas, 2017).

2.5 Fake News and the US 2016 Election

Following an investigation by BuzzFeed News relating to fake news and the 2016 US Elections, it was found that top-performing fake news stories on Facebook experienced higher user engagement than top stories from reputable news firms. A combined 8,711,000 shares, comments and user reactions were recorded from the top 20 fake news stories circulating on Facebook (Silverman, 2016). In contrast, top-performing online news stories, originating from reputable news firms, generated 7,367,000 shares, comments, and user reactions on Facebook (Silverman, 2016). A possible explanation to justify the mentioned figures could be attributed to the findings of Fernandez (2017) on the spread of fake news on social networking sites during the 2016 US Elections: people resonate with ideals they believe in, more than ideals they don't believe in (Fernandez, 2017).

An example of this spread is illustrated by Allcott & Gentzkov (2017): in July 2016, wtoe5news fabricated a story about Pope Francis endorsing Donald Trump as presidential candidate for the United States of America. What was not made clear in the presentation of the article is that wtoe5news is a self-proclaimed "fantasy news website.” Information about the site and its purpose were only placed on the site’s About page. The story received more than 1 million shares on Facebook (Allcott & Gentzkov, 2017).

Sydell (2016) reports on another example, in which NPR performed an extensive investigation into Denverguardian.com, a website that published a viral, false news story days before the election. The story, titled "FBI Agent Suspected In Hillary Email Leaks Found Dead In Apparent Murder-Suicide," was shared more than 500,000 times on Facebook (Sydell, 2016). The investigation uncovered several fake news websites that were created by the same person who authored the DenverGuardian.com story: NationalReport.net, USAToday.com.co, and WashingtonPost.com.co were hosted on a single server owned by Jestin Coller, CEO of Disinfomedia (Sydell, 2016).

2.6 Fake News and the Global Health Crisis COVID19

In December 2019, China informed the World Health Organization of an unknown pneumonia with a range of symptoms such as dry coughs, fever, and tiredness. The World Health Organization labelled the disease as COVID19 (Elhadad, Li, & Member, 2020). Since the start of the pandemic, social media has played its role in spreading misinformation about the virus. During the COVID19 pandemic, scientific and medical professionals have been faced with the challenge of individuals being able to create fake news content with ease (Scerri & Grech, 2020). The effects of such misinformation could lead to loss of trust in government, economic ramifications and persuasion into believing certain brands over others (Elhadad, Fun Li, & Gebali, 2020).

During the initial days of the pandemic, the medical community aided in the spread of misleading information by releasing inaccurate and contractionary information relating to COVID19 (Tagliabue, Galassi, & Mariani, 2020). Some medical professionals made statements to the media, often backed by no scientific evidence – the effects of such misinformation include the sudden hoarding of personal protective equipment (PPE) which in turn, led to limited PPE availability and a sudden rise in PPE pricing (Tagliabue, Galassi, & Mariani, 2020).

Governments around the world have taken measures against curbing the spread of misinformation related to the novel coronavirus (Rodrigues & Xu, 2020). In a South African provincial context, the Western Cape Government has compiled guidelines users can employ in determining the authenticity of COVID19 related news (Western

Cape Government, 2020). At a national level, the Government of South Africa has compiled a list of online fake news media that circulated in the country during the pandemic (South African Government, 2020). The fake news media items are a collection of images, short messages, and screenshots of fake documents. In Malta, researchers developed a website and Facebook page to deliver accurate, scientific information related to COVID19 (Scerri & Grech, 2020).

Presently, various organisations and researchers are collaborating in the fight against the spread of COVID19-related misinformation. From an information access perspective, the IEEE is providing free access to COVID19 research and standards on the IEEE Xplore Digital Library, in support of researchers working to tackle various spheres of the pandemic (Stickel & Tardo, 2020). Like the IEEE, SAGE Publishing is also offering free access to COVID19 related research (SAGE, 2020). In addition, the publisher compiled a document that lists all publications related to COVID19 (SAGE, 2020).

2.7 Fake News in South Africa

In recent years, South Africa has witnessed several cases of fake news and the effects that transpired from such cases. A notable example was that of a University of South Africa (UNISA) employee, who was exposed for running a fake news network. Following an extensive investigation, South African media firm, News24, found the UNISA employee, William Mahlatse Ramatseba, and his sibling authored multiple fake news websites and social media profiles (le Roux, 2018). The investigation also revealed Ramatseba used a single Google AdSense ID to generate revenue on 15 fake news websites and hid his WHOIS domain registration information on most domains (le Roux, 2018). MzantsiStories.com, MzantsiNewsOnline.com, GautengPraise.com and AllNews.co.za are a few of many fake news websites authored by Ramatseba (le Roux, 2018). Figure 2.1 illustrates the fake news network, as described in the text.

In a political context, Wasserman (2017) describes one particular case, where political debates sparked over the influence of the Gupta family and the former president of South Africa, Jacob Zuma. The premise of such debates emanates from serious allegations of corrupt deals between the family and associates, originating from a report authored by Thuli Madonsela, the former Public Protector of South Africa

(Wasserman, 2017). To sway public opinion, several fake Twitter accounts, which were found to be linked to media firms owned by the Guptas, and the Black First Land First (BLF) political party, were created (Wasserman, 2017).

A known fake news website, AllAfricaNews, published an article stating President Jacob Zuma resigned. News of this event triggered a positive response to the value of the South African Rand, until reports surfaced which refuted the claim made by AllAfricaNews (Kawa & Goko, 2018).

In addition to the mentioned key cases, the country had witnessed instances of alleged breaking news turning out to be a hoax; an example of this were reports claiming the former president of South Africa, Nelson Mandela, had passed away long before his death (Rodny-Gumede, 2018). In 2016, the ruling party, the African National Congress (ANC), had allegedly set up a 'War Room', which was intended to discredit opposition parties through the use of modern methods, such as buying Twitter accounts, use of social media influencers, and use of independent news websites and chat shows, without any links to the ruling party (Comrie, 2017).

Whilst South Africans debate the notion of nationalising the country's Reserve Bank, claims surfaced that a German citizen, Michael Duerr, owns 57.5% of the bank. Such claims were made by an Instagram post from Mzandile Masina, Mayor of Ekurhuleni (Clifford, 2019). After consulting the Reserve Bank, and studying the bank's shareholder index, Africa Check found that Michael Durr owned about 0.5% of the Reserve Bank's 2 million shares. (Clifford, 2019). Duerr further stated that he and his relatives own 12.5% of shares at the Reserve Bank, and not the 57% claimed on social media. The fact-verification organisation thus concluded this story as false. Figure 2.2 displays the Instagram post, which has since been deleted.

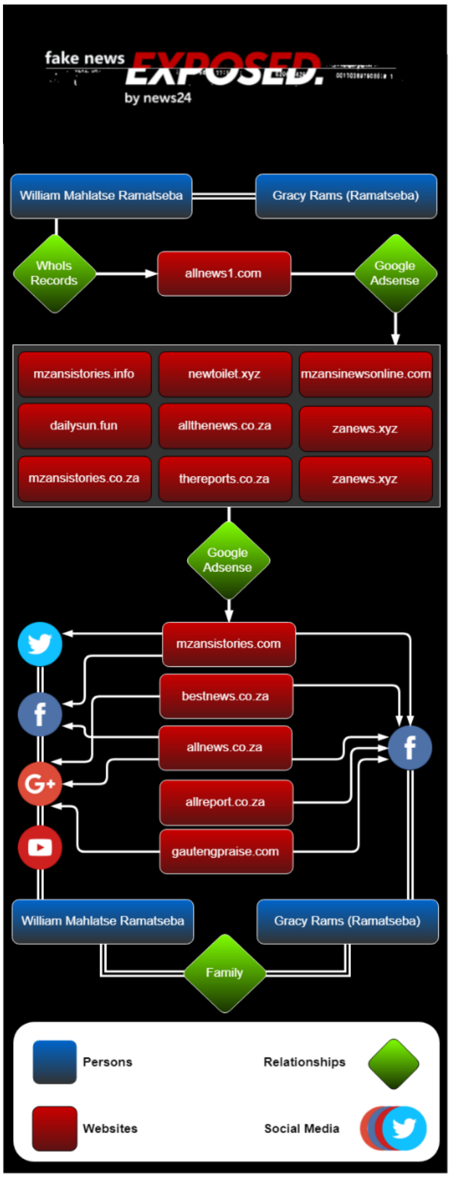
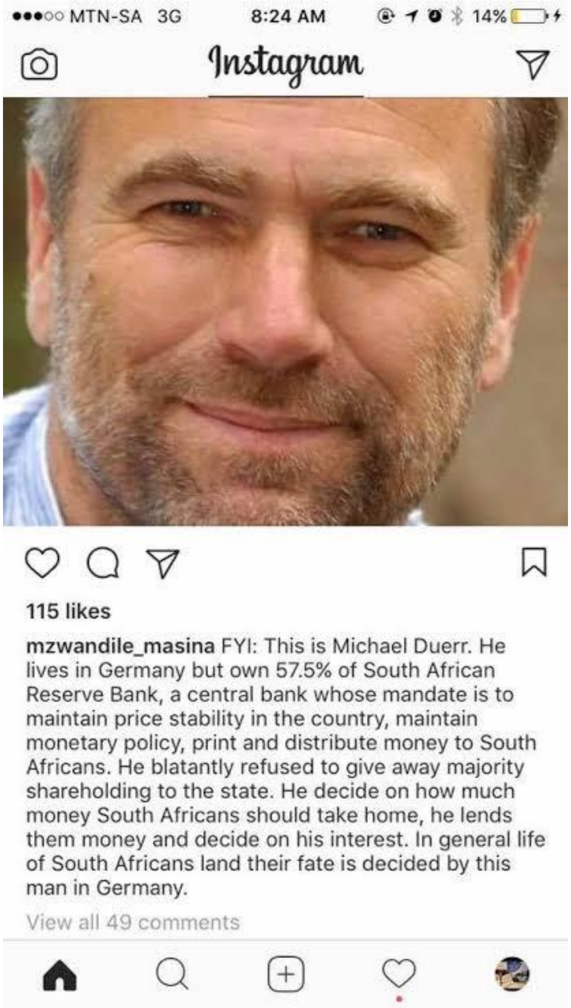


Figure 2.1: Fake news network exposed by News24 (le Roux, 2018).



115 likes

mzwandile_masina FYI: This is Michael Duerr. He lives in Germany but own 57.5% of South African Reserve Bank, a central bank whose mandate is to maintain price stability in the country, maintain monetary policy, print and distribute money to South Africans. He blatantly refused to give away majority shareholding to the state. He decide on how much money South Africans should take home, he lends them money and decide on his interest. In general life of South Africans land their fate is decided by this man in Germany.

View all 49 comments

Figure 2.2: Claim made by Mzwandile Masina in an Instagram post claiming Michael Duerr owns 57.5% of the South African Reserve Bank (Clifford, 2019).

2.8 Current Initiatives in Curbing the Spread of Fake News

With the proliferation of fake news, several strategies have been devised, and are being used or actively developed. The fight against fake news is a multi-disciplinary endeavour, involving a combination of social and computational solutions. In a social context, there has been a rise in fact-checking organisations. In a computational context, several researchers have shown machine learning technology could be a useful technology in fighting online misinformation. The first step in combatting fake news through the use of automated tools is understanding the fake news creator and the methods employed to spread the fake news content (Gravanis, Vakali, Diamantaras, & Karadais, 2019).

Online organisations, which verify the veracity of claims made by people and organisations, such as Snopes and PolitiFact, continue on an upward trajectory. Fact-checking organisations are led and managed by journalists who assess the credibility of a given news story and deliver judgement on whether a given story is true or false. In an African context, Africa Check was founded in London in 2012 and operates in Kenya, Nigeria, Senegal and South Africa (Africa Check, 2013). The African fact-checking organisation receives its support from many companies, with the Shuttleworth Foundation, Bill & Melinda Gates Foundation and the Luminate Group being the most notable supporters (Africa Check, 2013).

Fact-checking organisations have been received with mixed reviews. People generally do not question the credibility of information unless the information does not align with their ideals and beliefs, which in turn, could hinder the effectiveness of fact-checking organisations (Lazer, et al., 2018). Many fact-checking websites have been criticised for showing bias in their evaluation of news articles (Batchelor, 2017). In some cases, fact-checking might be fruitless – research has shown people tend to remember information and their emotions about a piece of information whilst, in most cases, forgetting the context behind the information. This, in turn, could increase a user's chance of accepting a given piece of information as truthful, even in a fact-checking context (Lazer, et al., 2018).

In a computational context, researchers have shown how machine learning techniques can address the problem of fake news. Machine Learning allows machines to solve problems through learning, and without manual programming (Stanescu, Mata-Toledo, & Gupta, 2018). Researchers have shown promising results, in fake news detection, by employing various supervised/semi-supervised machine learning algorithms with varying feature extraction techniques and data sets.

Over the years, rich data sets to aid research in online misinformation have emerged. Using fact-checked statements from Channel 4 and PolitiFact, Vlachos & Riedel (2014) constructed a dataset for misinformation detection, consisting of 221 statements, covering various topics in the United Kingdom and the United States. This dataset was later followed by Wang (2017) who created the LIAR dataset, consisting of 12386 short statements. Szpakowski (2018) released FakeNewsCorpus, a fake news dataset containing millions of articles, based on OpenSources' (n.d.) curated list of websites.

SignalMedia (2015) released a news dataset, consisting of 1 million articles, from various sources such as popular websites and blog articles.

Apart from mentioned solutions, education and training on assessing information credibility and quality have surfaced. Lazer et al. (2018) note such training might reduce an individual's view of the credibility of information on real news.

2.9 Indicators of Fake News

In a study aimed at understanding differences between fake online news articles and real online news articles from a writing style and language perspective, Horne & Adali (2017) note several characteristics of fake news articles, including longer headlines, shorter article bodies and the manner in which nouns, proper nouns and conjunctions are used (Horne & Adali, 2017).

Through the use of data analysis techniques, Yang et al. (2018) find that fake news articles tend to have a limited number of words whilst others have many words. The authors also add that real news articles generally have more sentences than fake news articles (Yang et al., 2018). Other observations include a higher number of question marks present on fake news articles than real news articles, and a higher number of capitalized words on fake news articles than real news articles (Yang et al., 2018). To calculate the similarity of the article headline to the article body, Wu, Cheng, & Chai (2017), Sreekumar & Chitturi (2019) and Masood & Aker (2018) calculate the cosine similiarity score. If the article headline and article body vectors are similar, a score of 1 is computed while a score of 0 is computed if the article headline and body vectors are not similar (Silva, Santos, Almeida, & Pardo, 2020). Owing to the literature covered in this chapter, some key indicators which can be used in the identification of text-based fake news, are:

1. Fake news articles generally contain short article bodies and longer article titles.
2. Fake news articles generally use simpler sentence structures.
3. Fake news article's generally have headlines which aren't related to the article body.

2.9.1 Fake News Articles Generally Contain Short Article Bodies and Longer Article Titles

Based on the observations noted by several authors in this chapter, this indicator considers the text lengths of article title's and bodies. In chapter 5, this indicator is mapped to appropriate features, such as the article headline length, and article body length.

2.9.2 Fake News Articles Generally use Simpler Sentence Structures

This indicator encapsulates several properties of the texts, such as the number of punctuation marks, the counts on the parts of speech present, the average sentence lengths, and the readability of the texts. The features which cover these properties are defined greater detail in chapter 5.

2.9.3 Fake News Article's Generally have Headlines Which are not Related to the Article Body

The relatedness of a given article to it's body has been examined by numerous authors. Chen, Conroy, & Rubin (2016) describe the "headline/body dissonance", where the headline of an article may not correlate with the article body. The cosine similiarity formula has been used by several authors to calculate the similiary of two texts. The results derived from this equation are included in the feature set and described further in chapter 5.

2.10 Conclusion

The fight against fake news will require collaborative efforts from stakeholders across multiple disciplines, namely the social sciences, journalism, computing sciences and technology fields. In addition, leading technology giants should contribute to reducing the spread of online misinformation. Fake news is useful to a given entity for many reasons; whether it is to generate revenue from advertising networks, sway public opinion into buying a narrative, or to bring forth financial or reputational repercussions to an individual or organisation. Confusion, mistrust, and financial damages are some of the many effects emanating from the phenomena.

This chapter provides coverage on online fake news and substantiates why online misinformation is a problem that needs investigation. This chapter also introduces some common differences that can be spotted in fake online news articles and real online news articles, which can be useful for computational solutions to the problem. These differences are transformed into a set of indicators, which are revisited in Chapter 5. The purpose of providing literature into this area is to address secondary objectives 1 and 2:

1. *Identify the motives behind the propagation of fake news.*

2. *Identify appropriate indicators, from literature, which could suggest an article is fake news.*

In this chapter, the challenges of tackling fake news are presented, from a technological and legislative perspective. The financial benefits of authoring fake news are also included in the body of literature. Chapter 3 provides extensive literature coverage of machine learning, and how the technology has been applied for automated solutions which combat online fake news.

3 Chapter 3 - Machine Learning in the Fight Against Fake News

3.1 Introduction

Machine Learning has seen a rise in popularity due to the rapid growth in computational capabilities and the greater availability of data (Stanescu, Mata-Toledo, & Gupta, 2018). The utilisation of this technology has solved various problems across multiple disciplines. In a business and commerce context, Khodabandehlou & Rahman (2017) evaluate several machine learning algorithms in predicting customers who are likely to end their relationship with one company for another company (defined as 'customer churning'). In a social networking context, Adewole & Anuar (2019) propose a framework for detecting spam messages and spam accounts, through experimenting with ten machine learning algorithms and presenting the best algorithms for spam message detection and spam account detection.

As noted by Adewole & Anuar (2019), machine learning uses two possible methods: supervised learning, where a classification model is built using sample training data, or unsupervised learning, where the algorithm learns from identifying patterns in the unlabelled data.

From a computational context, detecting fake news can be described as a multi-faceted task, with each task complementing one or more other tasks. Saquete, Tomás, Moreda, Martínez-Barco, & Palomar (2020) describe stance detection, deception detection, and polarity as sub-tasks that lead to the broader task of automated fake news detection. Stance detection involves examining the relationship between a given article's headline and body as well as people's sentiment on a given topic, whilst deception detection involves searching for certain cues or keywords in text (Saquete et al., 2020). Text polarity refers the human emotion (positive, neutral, or negative) contained in the text (Zhang & Ghorbani, 2019). Applying Natural Language Processing (NLP) techniques is a complex task due to the complex nature of human language. Although many machine learning algorithms exist, some of the frequently used machine learning models in the misinformation detection field include the Support Vector Machine, K-Nearest Neighbour and Random Forest. These algorithms are briefly introduced in sections 3.1.1, 3.1.2 and 3.1.3. To transform text to numeric feature spaces, where relationships between words are maintained, NLP models such as Word2Vec, Doc2Vec and GLoVe have been used by numerous researchers.

3.1.1 Support Vector Machine

The support vector machine is a popular machine learning model that can be used in linear and non-linear classification along with regression problems (Geron, 2017). In a classification task where data is labelled within two classes, a void space, referred to as the margin, separates the data into two classes (Manning, Raghavan, & Schütze, 2008). Data points that reside on the edge of the margins are referred to as the support vectors (Geron, 2017). The support vectors define the classification function, which determines where the separator is placed (Manning, Raghavan, & Schütze, 2008). Figure 3.1 illustrates a graphical representation of the Support Vector Machine, using data consisting of two classes, namely, circles and triangles.

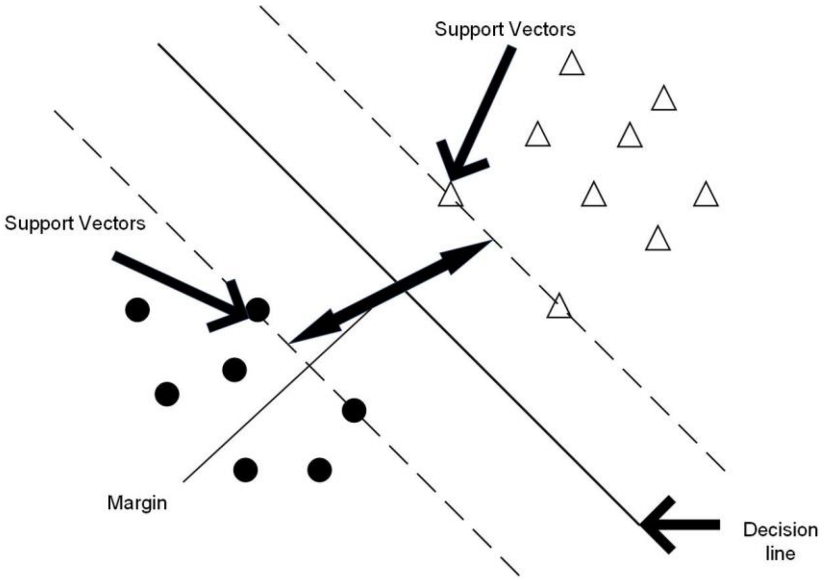


Figure 3.1: Support Vector Machine using data consisting of two classes (circles and triangles) (Manning, Raghavan, & Schütze, 2008).

3.1.2 K-Nearest Neighbour

The K-Nearest Neighbour (KNN) algorithm labels a data point to a class by determining the most dominant class by the number of samples present (Zhang Z. , 2016). Two key points relating to the algorithm are the function used to calculate the distance of a given data point, relative to other data points, and the k-parameter, which is used to define the neighbours (Zhang Z. , 2016). The Euclidian Distance formula is commonly used to calculate the distance of a data point, relative to other data points

(SciKit-Learn, 2012). Figure 3.2 illustrates the classification process employed by the K-Nearest neighbour algorithm.

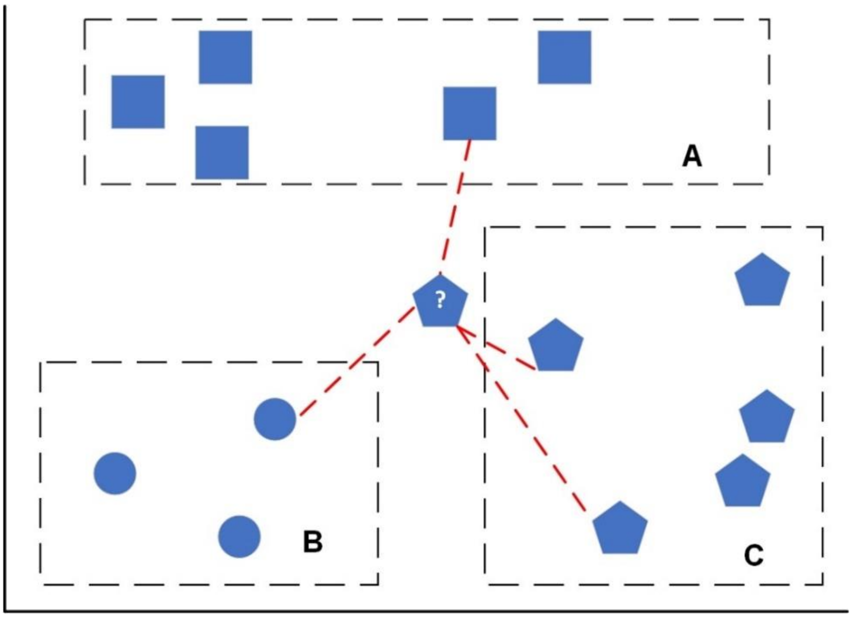


Figure 3.2: Graphical representation of the classification process employed by the K-Nearest Neighbour algorithm (Zhang Z., 2016)

3.1.3 Decision Trees

Decision Trees use a series of structured questions to determine the classification label for a given sample (Dangeti, 2017). A few advantages of using a Decision Tree to model a classification problem are the simplicity of the model and the low data preparation costs (SciKit-Learn, 2012). Figure 3.3 shows an example of a Decision Tree classifier. For illustration purposes, the maximum depth of the model is set to 2, and maximum features is set to 3. The fake real news dataset authored by Bisaillon (2020) is used.

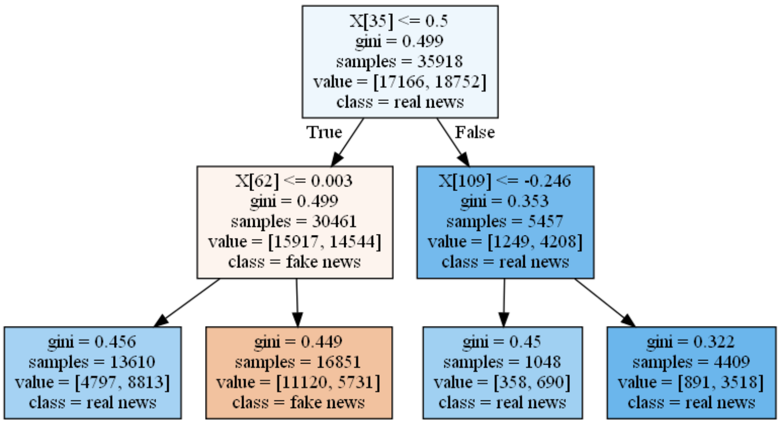


Figure 3.3: Graphical representation of a Decision Tree classifier.

3.1.4 Random Forest

The Random Forest classifier uses numerous Decision Tree models and averages the performance of all Decision Trees to optimize accuracy and reduce over-fitting (SciKit-Learn, 2020). At the end of each Decision Tree, voting determines the class of a sample (Dangeti, 2017). Figure 3.3 illustrates the composition of a Random Forest model.

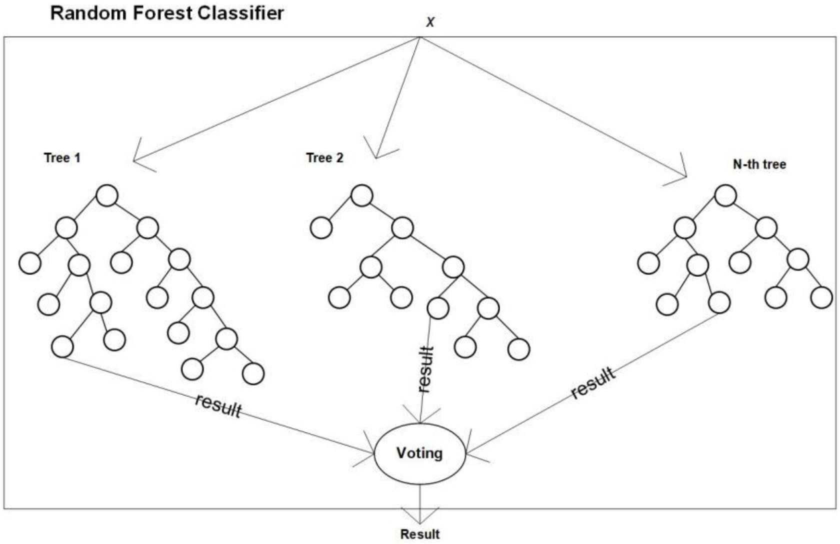


Figure 3.4: Overview of the Random Forest classifier and the collection of Decision Trees contained in the model (Dangeti, 2017)

3.2 Related Work: Automated Fake News Detection

In building a fact-checking system, Zhang & Ghorbani (2019) state that analysing the news content on its own is ineffective; such systems should include analysis of contextual information associated with an article, such as the author, the source, the target audience, and content (X. Zhang & Ghorbani, 2019).

Granskogen (2018) describes two approaches in the detection of fake news. The first, that being a linguistic approach, uses natural language processing to better understand the frequency of words, and the patterns present in the text. Examples of such techniques include Term Frequency Inverse Document Frequency (TF-IDF), sentiment analysis, and N-gram (Granskogen, 2018). Term Frequency-Inverse Document Frequency evaulates the importance of a word, relative to the document, and relative to the collection of documents (Granskogen, 2018). Sentiment analysis measures the polarity of the text (Granskogen, 2018). N-grams can be described as a selected sequencial subset of words or characters from a body of text. A unigram contains one value (or word), bigram contains two values, a tri-gram contains three words (Granskogen, 2018). The second approach, that being a contextual approach, takes into account the information surrounding the corpora, such as traffic, user relationships, links, and other relevant features (Granskogen, 2018).

Wu, Cheng, & Chai (2017) note how evaluating the stance – evaluating the relationship between the title and body of a given text – is an important indicator in detecting fake news. This coincides with the 'title/body dissonance', as described by Chen, Conroy, & Rubin (2016). Wu, Cheng, & Chai (2017) select 4 machine learning classifiers, namely, Support Vector Machine (SVM), Multinomial Naïve Bayes, Multilayer Perceptron (MLP), and softmax, for the task of determing the stance of an article. Stance detection is the process of determining the corrlation of a given article's title to the article's body (Wu, Cheng, & Chai, 2017). The authors calculate the relevance of an article title to its body by calculating the cosine similiarity for each article. In addition, the authors also select baseline features present in the dataset, provided by the FNC-1 task, which includes the a count on overlapping words, a count on the number of negative words, a count on overlapping n-grams, and a count on the polarity of article headlines and bodies (Wu, Cheng, & Chai, 2017). Polarity can be described as the number of negative words in a given text (Wu, Cheng, & Chai, 2017) . In the experiment, the multilayer perceptron classifier provides the best weighted accuracy among all classes, with an accuracy score of 77,74% (Wu, Cheng, & Chai, 2017). Similarly, Masood & Aker (2018) use the same dataset provided by the FNC-1 challenge to explore the detection of fake news articles by detecting stance of each model. Owing to the number of samples contained in each class (agree, disagree, discuss, related, unrelated) being imbalanced, the authors create two machine learning classification pipelines. The first pipeline, described as the 2-step classifier, has two L1-regularized logistic regision clasifiers, while the 2nd pipeline, described as the 3-steps classifier, has two L1 regularized logistic regression classifiers, and a random forest (RF) classifier (Masood & Aker, 2018). A detailed view of the 3-step classifier, using settings 1 and 2 is provided in Figure 3.5, Figure 3.6 and Figure 3.7.In the experiments, the authors find the 3-step classifier, using setting 2, delivers the best peformance, with an accuracy score of 89,18% (Masood & Aker, 2018). An overview of the 2-steps and 3-steps classifier is provided in Figure 3.5 and Figure 3.7.

In selecting features appropriate for the task of fake news detection, Choudhary & Arora (2020) justify the use of sentiment-related features on the premise that fake news authors typically embed a positive or negative stance in articles. The author also includes readability metrics to differentiate content written by professional journalists versus content written by fake news authors (Choudhary & Arora, 2020).

Hassan, Arslan, Li, & Tremayne (2017) developed ClaimBuster, a live fact-checking platform which leverages machine learning and natural language processing techniques to classify statements. The live fact checking platform contains several modules, such as the Claim Monitor, Claim Spotter, Claim Matcher, Claim Checker and Fact-Check Reporter (Hassan, et al., 2017). For the process of extracting features from the text, the authors use natural language processing technqiues such as obtaining sentiment, parts of speech tags, entity recgonition, and term frequency-inverse document frequency. A brief description of each module is provided below:

- The *Claim Monitor* module retrieves texts from a variety of sources such as live TV, Twitter and online websites (Hassan, et al., 2017).
- The *Claim Spotter* module computes a score between 0 and 1 for a given sentence. A score closest to 1 indicates the sentence contains check-worthy claims, while a score closest to 0 indicates the sentence contains non-factual, and subjective claims (Hassan, et al., 2017).
- The *Claim Matcher* module selects sentences identified by the Claim Monitor module, and searches a collection of fact-checks for matches (Hassan, et al., 2017).
- The *Claim Checker* selects appropriate evidence from the internet, supporting or negating a supplied claim (Hassan, et al., 2017).
- The *Fact-check Reporter* module generates a claim assessment report, including the supporting evidence and the computed claim spotter score. In addtion to delivering the report to the requesting user on Claim Buster’s website, the report is also published on the Claim Buster’s official accounts on various platforms such as Twitter and Slack (Hassan, et al., 2017).

Having noted the challenges associated with fake news dissemination on social media, Shu, Mahudeswaran, & Liu (2018) present FakeNewsTracker, a system for for fake news detection. In the system, the Social Article Fusion (SAF) model considers the social context and linguistic context in classifying fake news articles (Shu, Mahudeswaran, & Liu, 2018). The social context captures information related to user engagement on articles, such as the tweets and related replies (Shu, Mahudeswaran, & Liu, 2018). The linguistic context considers the contents of a given article. To capture linguistic context features, an Autoencoder network is selected. To capture social context, the authors use a Recurrent Neural Network, with Long Short Term Memory (LSTM) cells (Shu, Mahudeswaran, & Liu, 2018). Social context and linguistic feature vectors are concattenated for classification. The output layer of the model uses the softmax function to classify the articles (Shu, Mahudeswaran, & Liu, 2018). To test the efficiency of the model, the authors experiment with varying configurations of the SAF model. In the first setting, the model is configured to only consider the article contents.In the 2nd setting, the model is configured to consider only the social context, and in the 3rd setting, the model is configured to use both social and linguisitic contexts (Shu, Mahudeswaran, & Liu, 2018). Using accuracy, precision, recall and F1 score, the authors find the SAF model which considers both social and linguistic context features provides the best overall results for the BuzzFeed and PolitiFact datasets used (Shu, Mahudeswaran, & Liu, 2018).

Vishwakarma, Varshney, & Yadav (2019) propose a fake news detection system that examines both text and image. The process-flow employed examines a given image, extracts text from a given image, extracts entities from the extracted text, scrapes Google and identified links for related articles, then performs numerous operations, such as summarizing the content retrieved from the web, extract entities contained in the summarized text, and ensuring the entities extracted from images are related to the titles of web links (Vishwakarma, Varshney, & Yadav, 2019). The final step of the process is to classify an article as real or fake. If the Reality Parameter is greater than 40, the article is classified as real, otherwise, the article is classified as false (Vishwakarma, Varshney, & Yadav, 2019). For a given article, the Reality Parameter is calculated using equation 1 (Vishwakarma, Varshney, & Yadav, 2019):

$$Rp = \frac{Nrl}{Tnl} \cdot 100 \qquad (1)$$

Where:

- Nrl: Total number of reliable links
- Tnl: Total number links

Gravanis, Vakali, Diamantaras, & Karadais (2019) explore automated fake news detection through extensive research in several facets of the machine learning workflow. In the experiment, the authors define a pipeline which is selects the best feature combination. The best feature combination is provided to the machine learning algorithmn benchmark pipeline, where 6 machine learning algorithms, namely, Naïve Bayes, Support Vector Machine, Decision Tree, K-Nearest Neighbour, AdaBoost and Bagging are selected and evaluated. Following experimentation, the authors find the Support Vector Machine, and ensamble methods, such as AdaBoost and Bagging, provide the classification results (Gravanis, Vakali, Diamantaras, & Karadais, Behind the cues: A benchmarking study for fake news detection, 2019).

Zhang, Gupta, Kauten, Deokar, & Qin (2019) propose the FEND (Fake News Detection) framework, one which employs topic modelling and data clustering techniques in fake news detection. Using topic modelling, FEND can group data in a given dataset according to topic. FEND is able to determine whether a given article is fake or real by finding a relevant topic cluster. In the event an appropriate topic cluster is not found, the article is rendered fake. In cases where an appropriate topic cluster is found, the language structure of the article is compared against articles in the selected topic cluster. Should the article's structure poorly resemble those in the selected news cluster, the article is rendered false (C. Zhang et al., 2019).

Rubin, et al. (2019) build a news verification tool, LIT.RL, which can classify online news articles into three categories, namely, clickbait, satire fake news, and fake news. The tool uses the support vector machine algorithm in making inferences (Rubin, et al., 2019).

In a non-English context, Al-Ash & Wibowo (2018) study fake news detection using Term Frequency (TF) and Inverted Term Frequency (TF) to extract features from a collection of documents in Bahasa Indonesian (Al-Ash & Wibowo, 2018). The dataset used in this research is comprised of fake articles taken from https://turnbackhoax.id/ whilst data containing real news is taken from an Indonesian online news website, kompas.com (Al-Ash & Wibowo, 2018).

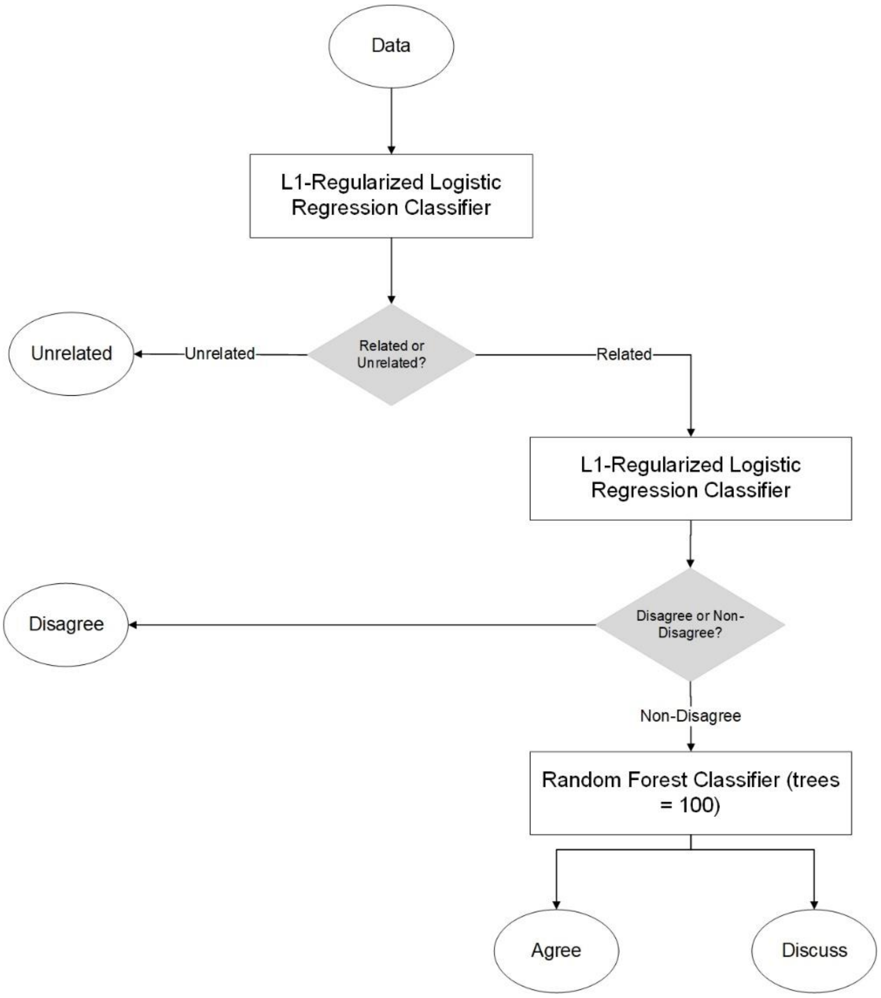

Data
L1-Regularized Logistic Regression Classifier
Related or Unrelated?
Unrelated
Unrelated
Related
L1-Regularized Logistic Regression Classifier
Disagree or Non-Disagree?
Disagree
Non-Disagree
Random Forest Classifier (trees = 100)
Agree
Discuss

Figure 3.5: Overview of the 3-step classifier, using setting 2 (Masood & Aker, 2018).

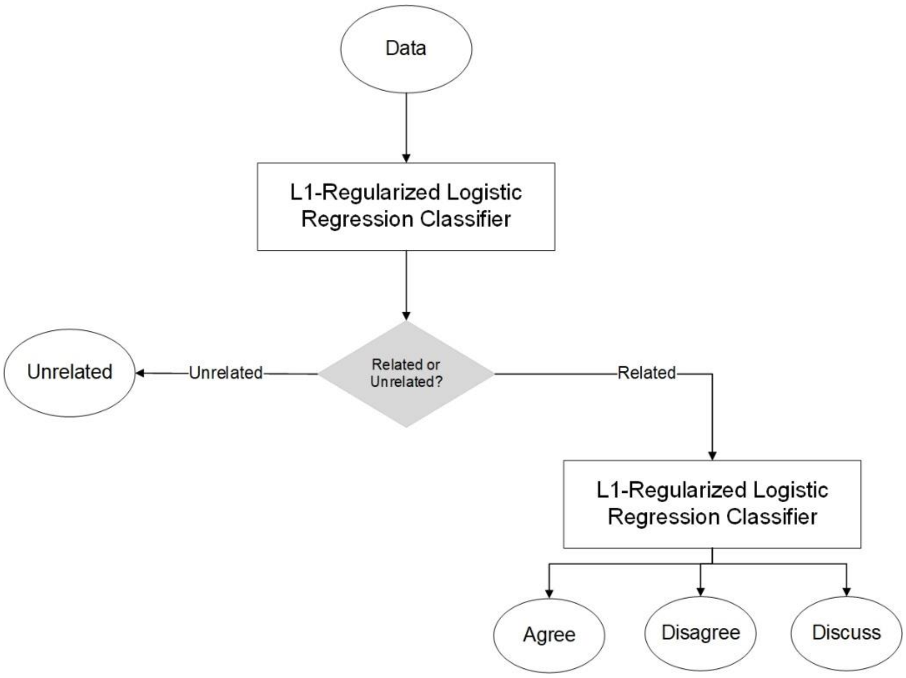

Data
L1-Regularized Logistic Regression Classifier
Related or Unrelated?
Unrelated
Unrelated
Related
L1-Regularized Logistic Regression Classifier
Agree
Disagree
Discuss

Figure 3.6: Overview of the 2-step classifier process (Masood & Aker, 2018).

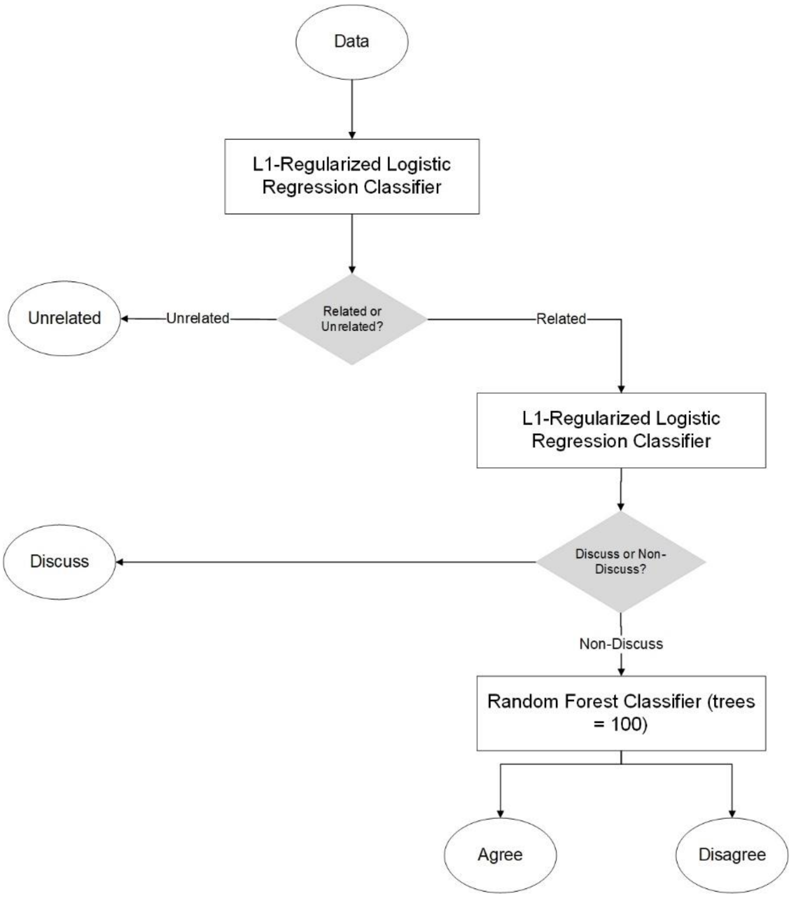

Data
L1-Regularized Logistic Regression Classifier
Unrelated
Unrelated
Related or Unrelated?
Related
L1-Regularized Logistic Regression Classifier
Discuss
Discuss or Non-Discuss?
Non-Discuss
Random Forest Classifier (trees = 100)
Agree
Disagree

Figure 3.7: Overview of the 3-step classification process, using setting 1 (Masood & Aker, 2018).

Similarly, in a Korean context, Kim & Jeong (2019) note an alternate approach to discerning false news from real news can be achieved by examining the article structure and comparing such article structures with other externally hosted articles. The authors propose a Korean fake news detection system that takes a given input, and compares the similarity of the input with articles contained in an article database (Fact DB) (Kim & Jeong, 2019). The authors propose a model which uses the Bi-Directional Multi-Perspective Matching for Natual Language Sentences (BiMPM), which has shown good results in sentence classification tasks (Kim & Jeong, 2019).

3.3 Datasets for Fake News Research

Publicly available datasets have been published and maintained by various organisations. Each of these datasets differs in terms of dataset size, topics and length of samples in the datasets. Rubin, Chen, & Conroy (2015) define nine conditions that should be fulfilled by any dataset to be applied in automated fake news detection. These requirements are:

1. *Availability of both truthful and deceptive instances*

2. *Digital textual format accessibility*
3. *Verifiability of 'ground truth'*
4. *Homogenous in lengths*
5. *Homogeneity in writing matter*
6. *Predefined timeframe*
7. *The manner of news delivery*
8. *Pragmatic concerns*
9. *Language and culture* (Rubin, Chen, & Conroy, 2015).

Predictive models should be able to find patterns in the data. Data should ideally be kept in text format, and items in the dataset should ideally come from credible sources, have appropriate text lengths, and touch on similar news topics (Rubin, Chen, & Conroy, 2015).

Datasets can be constructed in an automated or manual fashion; each workflow comes with its advantages and disadvantages. Though manual creation results in fine-tuned datasets, a notable drawback is that such datasets are prone to human error (Saquete et al., 2020). On the other side of the spectrum, automated datasets may not be of a high standard, but the methodologies employed to create such datasets offer flexibility (Saquete et al., 2020).

FakeNewsCorpus, created by Szpakowski (2018), is an open-source dataset of full news articles, constructed using a curated list of websites from OpenSources.co. Though this dataset is still in development, it currently contains 9,408,908 articles from 745 websites (Szpakowski, 2018). For each sample in the dataset, a tag, which identifies the type of article, is assigned. The tags assigned to samples in the dataset are fake, satire, bias, conspiracy, state, junk-sci (for junk science), hate, clickbait, unreliable, political and reliable (Szpakowski, 2018). SignalMedia (2015) released a news dataset, consisting of 1 million full news articles, from various news websites, published in the 1st to 30th September 2015 timeframe.

Ferreira & Vlachos (2016) use the Emergent dataset for the application of stance-based fake news detection. The dataset is based on the works of Silverman, (2015) which highlight the shortfalls and bad practices in current online media outlets and journalists. The labelled dataset contains 300 claims, which are linked to 2595 full news articles and related social media metrics. Each item in the dataset contains a label to describe its stance (for, against or observing) and a label to describe whether a given story was evaluated as false or true by journalists (veracity) (Ferreira & Vlachos, 2016).

Fake News Challenge (2016), a project aimed at tackling fake news through the exploration of artificial intelligence technologies, released the FNC-1 dataset as part of the first phase of their project, which focuses on stance detection. The dataset consists of 49 972 articles, of which are classified into four classes – agrees, disagrees, unrelated, and discusses (Misback & Pfeifer, 2017).

Other data collection techniques include scraping or crawling websites for content. In the FakeNewsTracker project, Shu, Mahudeswaran, & Liu (2019), devised a data collection strategy that entails collecting fake news articles from fact-checking websites like PolitiFact, then collecting related tweets and Twitter metrics related to the news articles from Twitter's API (Shu et al., 2019).

FakeNewsNet is a publicly available and regularly maintained data repository containing two datasets that provide news-related content, social engagement content and dynamic information (Shu, Mahudeswaran, Wang, Lee, & Liu, 2018). In the construction of the dataset, the PolitiFact and GossipCop web crawlers were employed in collecting labels associated with news articles. With regards to collecting social engagement data, the authors utilise Twitter's Advanced Search API, using news article headlines as the search query. Social engagement data collected include tweets, number of likes, reposts, user interaction and user profile metadata (Shu, Mahudeswaran, Wang, Lee, & Liu, 2018). Lastly, the dynamic aspect of the dataset includes regular updates to news content and social-related content – metrics associated with the dynamic context include timestamps of updates, and changes in user engagement (Shu, Mahudeswaran, Wang, Lee, & Liu, 2018).

Horne & Adali (2017) study fake news detection by using three datasets from three different sources, namely BuzzFeed, a self-authored dataset, and a dataset from (Burfoot & Baldwin, 2009), containing 233 satiric news articles and 4000 real news articles. In this study, Horne & Adali (2017) evaluate articles collected across the mentioned datasets, which can be categorized into one of three categories; namely fake news, real news, and satire news. In selecting articles for the respective datasets for the study, several filtering rules were applied to refine the first dataset; opinion-based articles and satirical articles were removed from the first dataset (Horne & Adali, 2017). The articles used in constructing the second dataset were based on the work of Zimdar (2016), who curated a list of known online fake news websites. The third dataset used in this study originates from the work of Burfoot & Baldwin (2009) which contains 223 satire news stories and 4000 real news stories (Horne & Adali, 2017).

Similarly, Rashkin, Choi, Jang, Volkova, & Choi (2018) used 13,995 full news articles from the English Gigaword dataset and crawled seven unreliable news websites in constructing a custom dataset for the fake news detection task.

Though there are numerous publicly available datasets for fake news detection in the English language, such availability is highly limited for other languages. In a study towards automated fake news detection in the Portuguese language, Silva, Santos, Almeida, & Pardo (2020) construct the FAKE.BR dataset for fake news detection, which comprises 3600 fake news articles and 3600 real news articles. The articles contained in the dataset are articles that were published between the January 2016 - February 2018 timeframe (Silva et al., 2020).

3.4 Datasets for COVID19 Misinformation

In light of the recent global health crisis, COVID19, Elhadad, Li, & Gebali (2021) construct the COVID-19-FAKES (Mohaddad, 2020) dataset, consisting of 3 263 464 tweets related to the global health crisis, in the English and Arabic languages (Elhadad et al., 2021). The authors collected COVID19-related tweets between 4 February 2020 and 10 March 2020, during the time of the disease's outbreak and the declaration of the global pandemic. The COVID-19-FAKES dataset is automatically labelled (real or misleading) using 13 machine learning algorithms, namely, Decision Tree (DT), K-Nearest Neighbour (KNN), Logistic Regression (LR), Linear Support Vector Machine (LSVM), Multinomial Naïve Bayes (MNB), Bernouli Naïve Bayes (BNB), Perceptron, Neural Network (NN), Ensamble Random Forest (ERF), Extreme Gradient Boosting (XGBoost), Bagging Meta-Estimator (BME), AdaBoost, and Gradient Boosting (GB) (Elhadad, Fun Li, & Gebali, 2020). The machine learning models are trained on tweets collected from official Twitter accounts and short statements verified by fact-checking organisations such as Snopes, PolitiFact and Poynter (Elhadad, Fun Li, & Gebali, 2020).

Hossain, et al. (2020) release the COVIDLIES dataset, which contains 6761 COVID19-related tweets. The dataset is built around 86 misconceptions about the virus. Each tweet is examined to determine if there is a match with any of the 86 misconceptions (Hossain, et al., 2020). In cases where a tweet does not match any misconception, the tweet is assigned the 'No Stance' label. For tweets matching any misconception, the tweet is further analysed to determine whether the tweet aligns with the misconception or contradicts the misconception. The classes 'Agree' and 'Disagree' are assigned respectively (Hossain, et al., 2020).

Patwa, et al. (2020) release a COVID19 dataset consisting of 10 700 fake and real news articles taken from social media sites like Facebook and Twitter, and also from fact verification websites like Snopes and PolitiFact. Using the dataset, the authors select 4 machine learning algorithms, namely, Decision Tree (DT), Logistic Regression (LR), Support Vector Machine (SVM) and Gradient Boost (GB) and find the Support Vector Machine classifier provides the best results, with an F1-score of 93.46% (Patwa, et al., 2020).

Banik (2020) released a 'COVID Fake News Data' dataset, which consists of 10 201 COVID-related claims that circulated on the internet. Claims are organised into two classes; namely 0 for fake, and 1 for truthful claims (Banik, 2020).

3.5 Fake News: Approaches to Preparing the Data

Upon collecting relevant data, the next step in the machine learning workflow is to ensure the data is ready for feature extraction. This process involves filtering out data for noise – removing abbreviations and removing words that add no value to the sentences are a few of the many strategies employed in cleaning the data.

In evaluating the effectiveness of various machine learning algorithms in fake news detection, Ahmed, Traore, & Saad (2017) employ various data pre-processing techniques commonly used. In the experiment, the authors use stop words removal, which is the process of discarding meaningless words in a text, such as conjunctions and pronouns (Ahmed et al., 2017). In addition, the authors employ stemming, a process that involves changing a word to its simplest form, to ensure the effective classification of text (Ahmed et al., 2017). In the process of cleaning articles contained in the curated dataset, Gilda (2017) removed article source names, social media handles and email addresses from a given articles' body (Gilda, 2017).

Al-Ash & Wibowo (2018) explore the applicability of natural language techniques, such as Term-Frequency (TF), and Term Frequency – Inverted Document Frequency (TF-IDF) for the detection of fake news. The authors use a collection of online news articles, in the Indonesian Bahasa language, and the Support Vector Machine algorithm for the task of detecting fake news (Al-Ash & Wibowo, 2018). The authors conclude the study by stating Term-Frequency is a feature that can be used to differentiate fake news articles from real news articles, owing to the 96,74% performance score (Al-Ash & Wibowo, 2018).

3.6 Fake News: Approaches to Extracting Features

The next process in the machine learning workflow is to extract meaningful features from the data, following data pre-processing techniques. Feature Extraction involves transforming data into meaningful features (in the form of vectors) that a machine learning algorithm can interpret (Stanescu, Mata-Toledo, & Gupta, 2018). Researchers in the fake news realm have extracted various features for various, logical reasons.

Ahmed et al. (2017) explore two feature extraction techniques, namely Term Frequency (TF) and Term Frequency – Inverse Document Frequency (TF-IDF). Term Frequency counts the occurrence of words in a document. Following the word count, each identified word and its count (vector) are normalized in such a manner that all elements add up to 1 (Ahmed et al., 2017). TF-IDF considers the importance of a word in a collection of documents in the dataset (Ahmed et al., 2017). Equations 2 (Tanvir, Mahir, Akhter, & Huq, 2019) and Equation 3 (SciKit-Learn, 2012) show the mathematical calculations used to calculate TF and IDF features:

$$tf(t) = \left(\frac{Number\ of\ times\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}\right) \tag{2}$$

$$idf(t) = \log\left(\frac{n}{df(t)+1}\right) \tag{3}$$

Where:

df(t): The document frequency of term *t*

n: number of documents in the collection of documents

tf(t): term frequenct of term *t*

idf(t): iverted document frequency for term *t*

In the ClaimBuster project, Hassan et al., (2017) extract multiple features from the data. These include extracting sentiment scores, between -1 and 1, as well as extracting TF / TF-IDF features, Parts of Speech (POS) tags and Entity Types (Hassan et al., 2017).

Mikolov, Sutskever, Chen, Corrado, & Dean (2013) introduce word2vec, a neural network model which generates word embeddings in a given set of words. Le & Mikolov (2014) highlight that simple document vector extraction techniques, such as the bag-of-words, do not capture the order of words and meaning for words and sentences. The relationships between words in a vector matrix produced by the word2vec model can be represented in a series of simple arithmetic expressions (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). Le & Mikolov (2014) present the paragraph vector framework, where each paragraph and word contained in a document, are assigned a vector. The next word is predicted by concatenating the paragraph and word vectors (Le & Mikolov, 2014).

Horne & Adali (2017) select features that assist in understanding the differences between fake news articles and real news articles, in three areas: from a sentence style perspective, sentence complexity perspective and psychological perspective. The authors employ Parts of Speech (POS) tagging, and a count on several grammatical indicators (punctuation, capital letters, etc.) to describe the structure of a sentence. To comprehend the complexity of words used, the authors calculate the readability of articles using three grading systems, namely, Gunning Fog, SMOG Grade and Flesh-Kincaid (Horne & Adali, 2017). The authors also calculate the Type-Token Ratio for each document, which is a ratio of distinct words divided by the total words in a document. Finally, to obtain features that describe the psychological context of the articles, the authors use a sentiment analysis tool to measure the emotion for each document (Horne & Adali, 2017).

Several authors employ cosine similarity to calculate the distance between vectors belonging to the text. Similar texts will have a higher cosine similarity score, whilst unrelated texts will have a low cosine similarity score. Wu, Cheng, & Chai (2017) employ cosine similarity to determine the similarity of an article's title and body. The cosine similarity produces a score between 0 and 1, where 0 indicates low relativity and 1 represents high relativity (Silva et al., 2020).

In the context of stance detection, Ferreira & Vlachos (2016) extract features from a given article title and article body. From the headline, the author employs the Bag-of-Words and RootDist techniques for extracting features.

Various models for text summarization have been developed and utilised as a means of extracting features from the text. Text summarisation exists in two categories, namely, abstractive text summarisation and extractive text summarisation (Liu, 2019). Abstractive summaries paraphrase the original text, whilst extractive text summarization results in a summary that contains the most important words from the original text (Liu, 2019).

3.7 Transfer Learning in Machine Learning Tasks

In the field of machine learning, transfer learning is increasingly gaining popularity due to the promising results the technique yields. In a natural language processing context, transfer learning is the process of pre-training a model, which can be used for other tasks in a machine learning workflow (Raffel et al., 2019). Over the years, several natural language processing frameworks, and pre-trained models, have been developed for use by the research and business communities. Researchers at Facebook introduce the InferSent model, which generates sentence embeddings, which can be used for other supervised learning tasks (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2018). The encoder is based on Bidirectional Long Short Term Memory (BiLSTM) architecture with max-pooling (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2018). The authors train the model using the labelled Sandford Natural Language Inference (SNLI) dataset, consisting of 570 000 sentence pairs and 3 labels, namely, entailment, contradiction, and neutral (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2018). The Natural Language Inference task employed by Conneau, Kiela, Schwenk, Barrault, & Bordes (2018) aims to understand the relationship between sentences in texts. Using 12 NLP transfer tasks to test the efficacy of the sentence embeddings. The BiLSTM model provides superior results, compared to the other models, and further provides state-of-the-art results for 2 of the transfer tasks, namely SICK-R and SICK-E (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2018). Given the known best sentence encoding model, SkipThought-LN, the authors find their model provides superior results compared to the SkipThought-LN model, whilst being trained on less data, and taking less time to train (Conneau, Kiela, Schwenk, Barrault, & Bordes, 2018).

Wolf et al. (2019) release the Python transformers library, which provides simple-to-use APIs that cover a variety of natural language processing tasks, such as text generation, text classification, text summarization, text translation, etc. (Wolf et al., 2019). The library allows developers to provide a pre-trained model, which is downloaded and used for a given natural language processing task. The models implemented in the library consist of three key components: namely, a tokenizer, a transformer, and a head. The tokenizer transforms text into numeric, index values. The transformer creates contextual text embeddings by processing index values produced by the tokenizer. Finally, the head is responsible for producing a prediction using the contextual embeddings produced by the transformer (Wolf et al., 2019).

Several authors have used the outputs generated from a document and word embedding models, such as word2vec and doc2vec as part of inputs for classification problems. Wang (2017) uses a pre-trained word2vec model to generate text embeddings for their proposed hybrid neural network architecture, aimed at fake news detection. Ngada & Haskins (2020) use the doc2vec model to generate document embeddings as part of inputs for the task of fake news detection using several machine learning algorithms, namely K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Random Forest (RF), Extreme Gradient Boost (XGBoost), Random Forest (RF), AdaBoost (AB) and the Decision Tree (DT).

3.8 Transformers and Natural Language Processing

Transformer architecture has brought a disruptive change in the Natural Language Processing field. Structurally, the Transformer model consists of an encoder and a decoder. The encoder transforms a given input into an output. The decoder transforms the output generated by the encoder and generates outputs one element at a time (Vaswani et al., 2017). The decoder portion of the transformer architecture uses previously generated outputs as part of its inputs when generating subsequent outputs (Vaswani, et al., 2017). In the encoder and decoder, the transformer includes self-attention and feed-forward layers. Both the encoder and decoder consist of six layers and two sub-layers. A few popular transformer architectures include the Bidirectional

Encoder Representations from Transformers (BERT), Generative Pre-trained Transformer (GPT) and the Text-To-Text Transfer Transformer (T5). A crucial component of the Transformer architecture is the self-attention layers, which is briefly discussed in section 3.8.1.

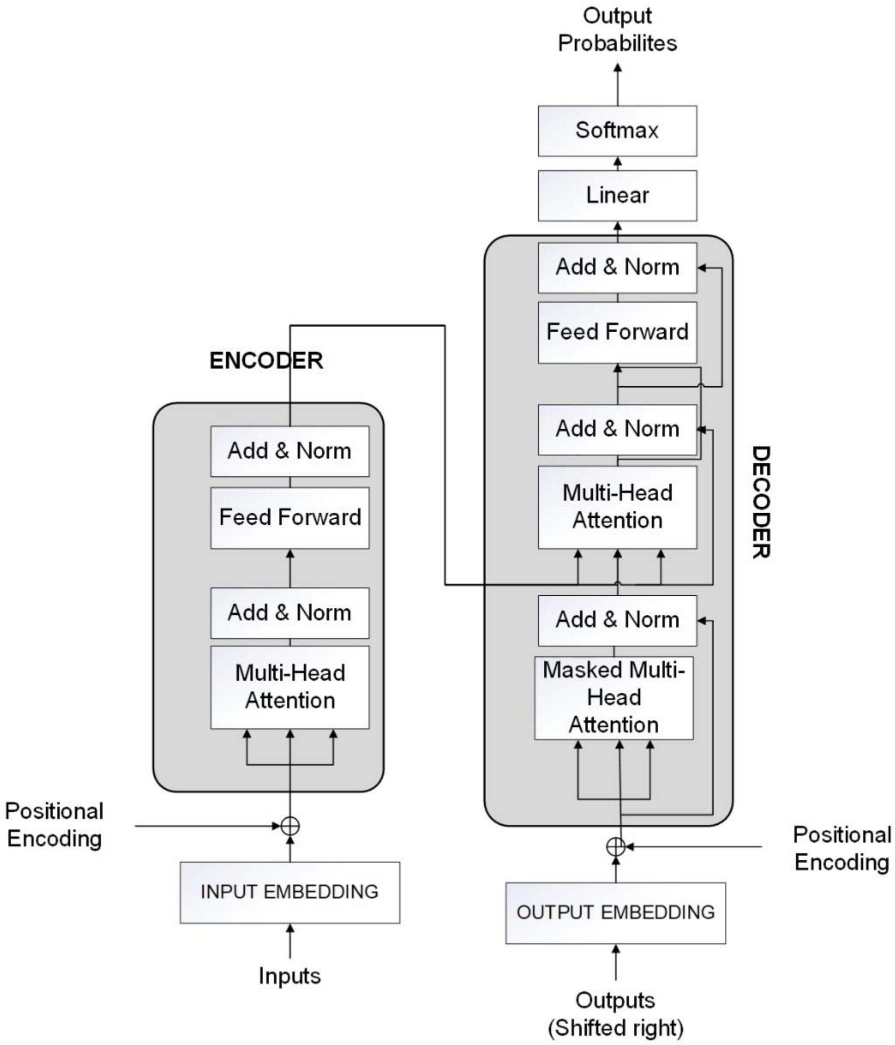


Figure 3.8: Overview of Transformer architecture (Vaswani et al., 2017)

3.8.1 Self-Attention in Transformer Architecture

The transformer architecture uses a multi-head attention layer. The attention function associates a query, keys and values to output. Each value is assigned a weight, which is calculated through the use of a compatibility function, using the query and key (Vaswani, et al., 2017). In the multi-head attention layer, the queries, keys and values are projected a certain number of times, with each projecting containing different linearly projected keys and values. The Scaled Dot-Product Attention function is applied in parallel, to each version of the queries, keys and values (Vaswani, et al., 2017). If Q represents the query, K represents the keys, V represents the values, and d_k represents the dimension of the keys and queries, the attention function is computed using equation 4 (Vaswani, et al., 2017):

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

Therefore, the Multi-Head Attention function can be represented, using equation 5 (Vaswani, et al., 2017). W_i^Q, W_i^K, W_i^V and W^O represent the linear projections of the queries, keys and values (Vaswani, et al., 2017).

$$\begin{gathered} MultiHead(Q, K, V) = Concat(head_1 \dots head_h)W^O \\ where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{gathered} \tag{5}$$

3.8.2 Generative Pre-Trained Transformer (GPT)

(Radford, Narasimhan, Salimans, & Sutskever (2018) introduce the Generative Pre-Trained Transformer (GPT) model. The model employs the decoder portion of the Transformer architecture. In the creation of the GPT model, which uses transformer architecture, Radford, Narasimhan, Salimans, & Sutskever (2018) state two reasons for selecting transformer architecture. The first is the model's successes in numerous natural language tasks, such as machine translation, document generation and syntactic parsing (Radford, Narasimhan, Salimans, & Sutskever, 2018). The second is the model's ability to handle long term-dependencies better, compared to recurrent neural networks, due to the availabilty of more memory (Radford, Narasimhan, Salimans, & Sutskever, 2018).

3.8.3 Bidirectional Encoder Representations from Transformers (BERT)

Devlin, Chang, Lee, & Toutanova (2019) introduce the Bidirectional Encoder Representations from Transforms (BERT) model, which is also based on the Transformer architecture. The authors state 2 advantages of the BERT model over previous works. The first advantage is the model employs a single architecture for pre-training and fine-tuning tasks (Devlin, Chang, Lee, & Toutanova, 2019). The second advantage is BERT's use of a bidirectional transformer architecture. BERT uses a Masked Language Model (MLM) and the Next Sentence Prediction (NSP) to pre-train the bidirectional Transformer (Devlin, Chang, Lee, & Toutanova, 2019). In the MLM model, tokens are randomly masked. The objective of the MLM is to predict the masked word, by using the word's context. The left-to-right and right-to-left context's of the masked word are concatenated (Devlin, Chang, Lee, & Toutanova, 2019). Through experimentation, using the BERT-BASE and BERT-LARGE models, the authors find the BERT model provides superior performance compared to other language models such as GPT, ELMo and SOTA in 11 natural language processing benchmark tests (Devlin, Chang, Lee, & Toutanova, 2019).

3.8.4 Text-To-Text Transfer Transformer (T5)

Raffel, et al. (2020) introduce the Text-To-Text Transfer Transformer (T5) model. The T5 model accepts text as inputs and provides text as outputs (Raffel, et al., 2020). The T5 model provides a single model for a range of natural language processing tasks such as machine translation, question answering, text summarization and text classification (Roberts & Raffel, 2020). An overview of the model is illustrated in Figure 3.9.

The transformers Python-based library, released by (Hugging Face, 2020) provides access to several pre-trained natural language processing models, such as T5, BERT, etc. To action a specific task for the T5 model, such as text translation or summarization, the text must be pre-appended with the appropriate task. To translate text from English to German, text must be pre-appended with *'translate English to French: '* and to summarize text, the text must be pre-appended with the 'summarize: ' keywords (Hugging Face, 2020).

3.8.5 Related Work: Transformers in Fake News Detection

From a COVID-19 perspective, Chen, et al. (2021) propose the Robust-COVID-Twitter-BERT (Ro-CT-BERT) model for the detection of COVID-19 misinformation. The model combines CT-BERT and RoBERTa transformer architecture. In this work, the authors argue the performance of classification models for COVID fake news detection can be improved by expanding the models' dictionary with terms specific to COVID19, providing a heated softmax function so the model can pay more attention to COVID19 specific terms, and employ adversarial training on the model to improve the models' generalization capabilities (Chen, et al., 2021). Using several variants of the BERT, RoBERTa, and ALBERT models, and the CT-BERT model for comparison, the Ro-CT-BERTA model outperforms all selected models, with an accuracy score of 99,01%, precision of 99,02%, recall of 99,01% and F1-Score of 99,01%.

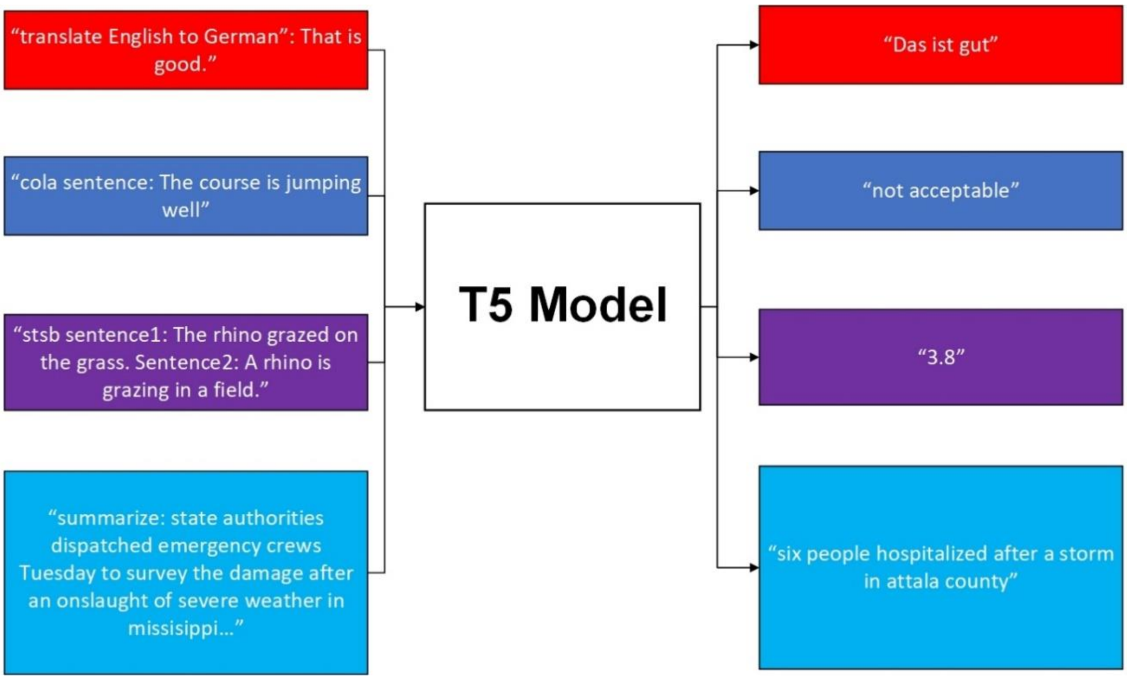

"translate English to German": That is good."
"cola sentence: The course is jumping well"
"stsb sentence1: The rhino grazed on the grass. Sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews Tuesday to survey the damage after an onslaught of severe weather in missisippi…"
T5 Model
"Das ist gut"
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county"

Figure 3.9: Overview of the T5 model and several tasks the model can process (Raffel, et al., 2020).

Using two collections, containing fake and real news articles, authored by (Shu, Mahudeswaran, Wang, Lee, & Liu (2018), Kula, Kozik, Choras, & Wozniak (2021) investigate the application of transformer neural network-based methods for fake news detection, using the Flair Python-based library. To evaluate the performance of the selected transformer models, namely, xlNET-LARGE-CASED, RoBERTA-LARGE, DistilGPT2, BERT-LARGE-CASE-TDE, BERT-LARGE-CASED-DRE, DistilBERT-BASE-UNCASED, the authors select performance metrics precision, recall and F1 Score (Kula, Choras, Kozik, Ksieniewicz, & Wozniak, 2020).

(Kula, Kozik, & Choras (2021) present 2 hybrid models for the task of fake news detection, which employ BERT, RoBERTa and RNN architecture. The authors use two datasets, namely the ISOT dataset authored by (Bisaillon, ISOT Fake News Dataset, 2020)

Using the FNC-1 dataset, which was used in the Kaggle competition *Fake News Challenge* (stage 1) Slovikovskaya & Attardi (2020) revisit the fake news stance detection problem, through the use of 3 Transformer based architectures such as BERT, XLNet and RoBERTa. Using the featMLP model, authored by (Hanselowski, PVS, Schiller, & Caspelherr, 2017) as the baseline for experimentation, the authors find the transformer-based models BERT, XLNet and RoBERTa provide performance gains in the region of 8% to 20% for the 'related' class classification (Slovikovskaya &

Attardi, 2020). In addition, BERT, XLNet and RoBERTa provided improved precision, recall and F1-Scores compared to the baseline featMLP model (Slovikovskaya & Attardi, 2020). In the Fake News Challenge – 1 competition, the featMLP model outperformed all other models for stance detection (Slovikovskaya & Attardi, 2020).

3.9 Machine Learning Algorithms for Fake News Detection

Several machine learning algorithms are available and can be applied in many topics. The underlying data supplied to such algorithms can originate from data that exists in many forms, such as textual data, images, audio and so forth. Extensive research has been carried out on analysing and solving the problem of text-based fake news, using supervised and unsupervised learning approaches.

In the ClaimBuster project, Hassan et al. (2017) use nine machine learning algorithms with three classes present in the underlying dataset (NFS – Non-Factual Sentence, UFS – Unimportant Factual Sentence, CFS – Check-worthy factual sentence). Using varying combinations of features, the authors found the Support Vector Machine (SVM) classifier to yield the best accuracy overall. The Support Vector Machine algorithm is a classification algorithm that separates its classes with a hyperplane, a defined space between the classes represented (Bondielli & Marcelloni, 2019). When increasing the dataset size, SVM's performance was not affected, whilst the Naïve Bayes Classifier (NBC) experienced performance improvements (Hassan et al., 2017).

Ahmed et al. (2017) experiment with six machine learning algorithms, namely Stochastic Gradient Descent (SGD), Support Vector Machine, Linear Support Vector Machine (LSVM), K-Nearest Neighbour (KNN) and Decision Trees (DT). The experiment examines feature extraction techniques TF-IDF and TF, changes to feature sets, and changes to the n-gram (Ahmed et al., 2017). In the experiment, the authors make several observations: the linear classifiers used in the experiment returned better results than the non-linear classifiers, and with an increase to the n-gram, the machine learning algorithms experience a drop in accuracy. The resultant model in this experiment attains an accuracy rate of 92%, using LSVM and unigram features (Ahmed et al., 2017).

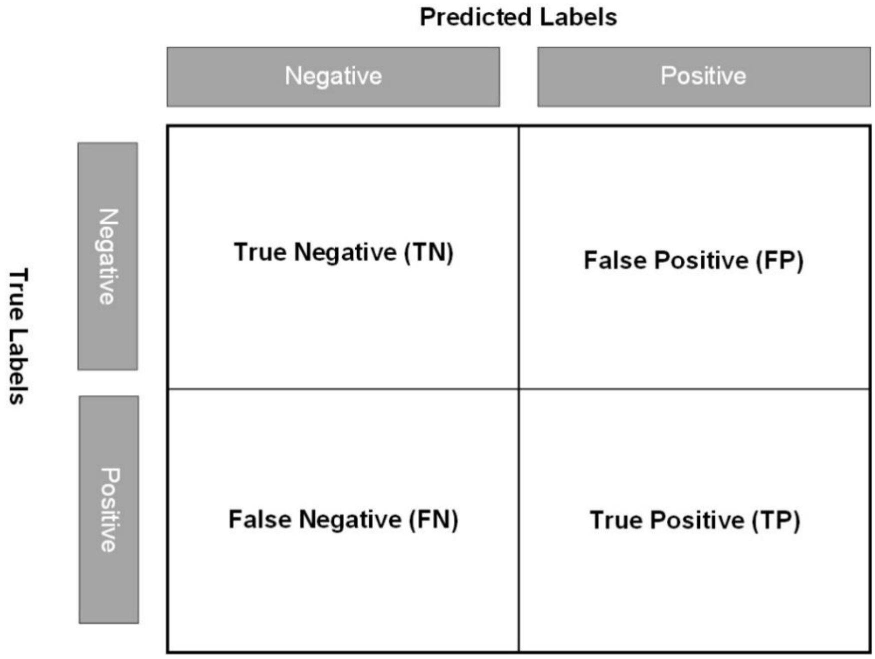
Machine Learning algorithms have several parameters which can influence the overall classification performance of a machine learning model. The majority of machine learning frameworks provide systematic approaches in determining the optimal configuration for such models. The hyperparameter selection process determines the best configuration for a given classifier, after examining all possible configurations. Two of the most common hyperparameter selection techniques are grid search and randomised search (SciKit Learn, 2012). Grid search examines all combinations of provided parameters whilst randomised search selects several parameters from a given collection of parameters (SciKit Learn, 2012). Such techniques have been used by researchers in the fake news realm. Wang (2017) employs grid search to establish optimal parameters for machine learning algorithms Logistic Regression and Support Vector Machine respectively. Similarly, Silva et al. (2020) use the grid search technique to find optimal parameters for SVM, RF, Bagging and AdaBoost. Silva et al. (2020) note that the performance of models is dependent on the configuration of the provided parameters.

3.10 Evaluating Machine Learning Algorithm Performance

Part of the machine learning workflow is ensuring the built models work as expected and can make accurate predictions. Apart from observing results after the machine learning algorithm has been built, several methods have been developed to evaluate the performance of a machine learning algorithm.

With most machine learning workflows, data is split between a training portion and a testing portion. The machine learning algorithm is trained using the training portion, and the model is tested using the test portion of the data (Vierra, Pinaya, & Mechelli, 2019).

The confusion matrix is a table that measures a machine learning model's ability to make accurate predictions (Adewole & Anuar, 2019). Figure 3.5 represents the confusion matrix model used to quantify a classification algorithm's performance.



		Predicted Labels	
		Negative	Positive
True Labels	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

Figure 3.10: Graphical representation of the confusion matrix model (Vierra, Pinaya, & Mechelli, 2019).

The confusion matrix can be used to calculate four measures, namely, accuracy, precision, recall and the F-Measure score (Deng, Liu, Deng, & Mahadevan, 2016). Khodabandehlou & Rahman (2017) defines the four performance-related equations:

$$Accuracy = \frac{TP + TN}{(TP + FP + TN + FN)} \tag{6}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{7}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{8}$$

$$F - Measure = \frac{2\ x\ (Precision\ x\ Recall)}{Precision + Recall} \tag{9}$$

Where:

- True Positive (TP) = Instances that are correctly labelled as true.
- False Positive (FP) = Instances that are falsely labelled as true.
- False Negative (FN) = Truthful instances which are falsely labelled as false.
- True Negative (TN) = Instances that are correctly labelled as false.

The accuracy metric determines the overall number of samples correctly identified as truthful, relative to the provided dataset (Brownlee, 2014). Recall measures a classification models' ability to correctly classify all known truthful samples in a dataset

(SciKit-Learn, 2021). Precision is a metric that measures a classification models' ability not to misclassify a truthful sample (SciKit-Learn, 2021). F-Measure is a metric that calculates the average of a classification models' precision and recall. A good score is a figure close to 1, while a bad score is a figure close to 0 (SciKit-Learn, 2021).

The k-fold cross-validation methodology is a popular validation methodology. In 10-fold cross-validation, the dataset is divided into 10 subsets. At each iteration, one of the subsets are used as the testing dataset, whilst the remaining nine are used as the training data for the machine learning classifier (Adewole & Anuar, 2019). Figure 3.2 illustrates 10-fold cross-validation. At each iteration, a different portion of the dataset (shaded in black), is selected as the test data, whilst the remaining data is used as training data. The overall performance result is calculated by taking aggregating performance results at all iterations (Vierra, Pinaya, & Mechelli, 2019).

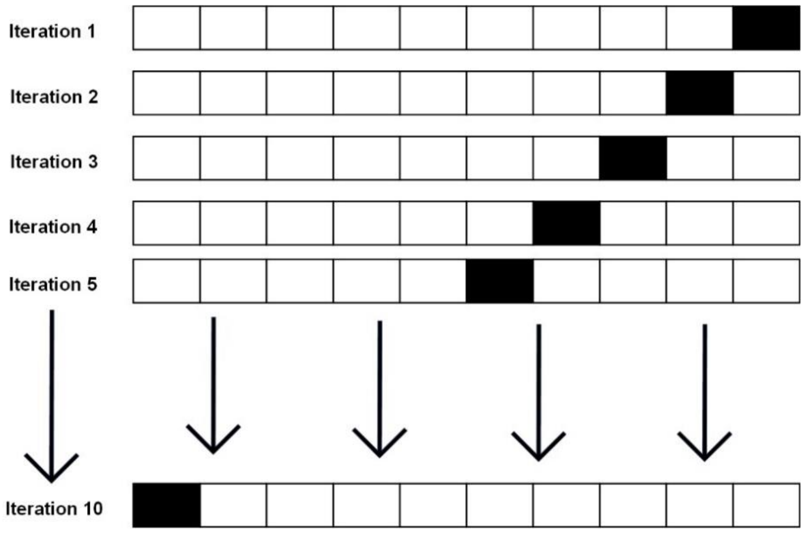


Figure 3.11: Example illustration of 10-fold cross-validation (Vierra, Pinaya, & Mechelli, 2019).

In scenarios where a strong imbalance exists between classes in a dataset, the stratified k-fold cross-validation technique can be used. The stratified k-fold cross-validation technique ensures the imbalanced dataset is split in such a manner that the same class distributions are maintained at each split (Brownlee, 2020). Pathak & Srihari (2019) select the stratified k-fold cross-validation technique to evaluate classification performance on the selected Bi-Directional LSTM neural network model, for fake news detection.

3.11 Deep Learning in Fake News Detection

Deep learning, a subset of machine learning, is an artificial intelligence-based function that resembles much of its behaviours from the human mind. Deep learning involves the construction of artificial neural networks, where data flows through layers of the neural network. The basic anatomy of a neural network consists of multiple hidden layers, and each layer contains multiple, interconnected nodes, with each node having an activation function (Sharma, Sharma, & Athaiya, 2020). In the field of fake news detection, neural network-based methodologies have been examined by researchers in the AI community. The performance of a neural network is dependent on the number of hidden layers, the selected activation function, training methodologies, and selected hyperparameters (Sharma et al., 2020). Neural networks accept numeric inputs and perform a series of arithmetic operations to optimise network parameters (Nasir, Khan, & Varlamis, 2021). The most popular neural network architectures for natural language processing are Recurrent Neural Networks (RNN) and Convolution Neural Networks (CNN) (Deepak & Chitturi, 2020). From a high-level view, the basic anatomy of an artificial neural network is illustrated below:

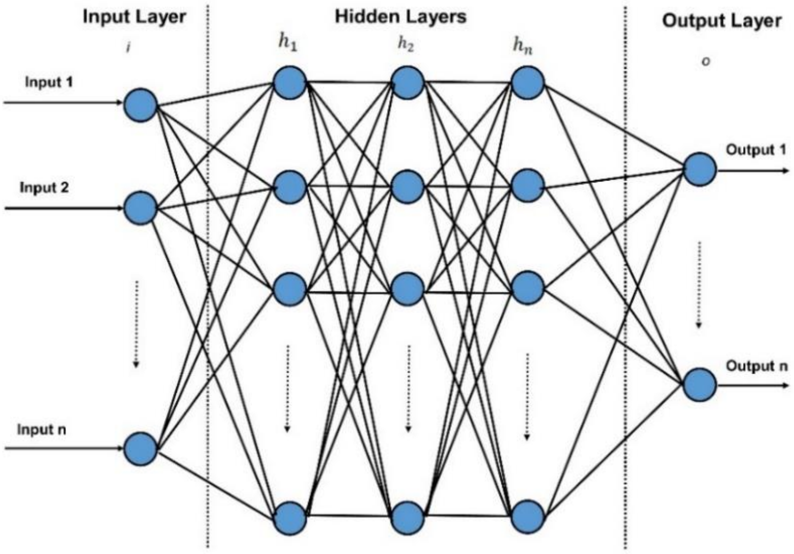


Figure 3.12: Basic Artificial Neural Network Architecture (Sharma et al., 2020)

Artificial Neural networks learn through the process of back-propagation. In these techniques, the cost function is responsible for updating weights at each of the networks' layers (Cuevas, Kaliyah, Goswami, Narang, & Sinha, 2020). Activation functions in neural networks generate an output by taking inputs and weights and applying an applicable function (Sharma et al., 2020). The output is sent to the next layer in the neural network.

Frequently used activation functions include the Rectified Linear Unit (ReLU), tanh, and sigmoid. ReLU is a commonly used activation function for neural networks, and its main function is to replace negative values in its input with zeroes in the network (Cuevas et al., 2020). One notable advantage of the ReLU activation function is certain neurons are active; only neurons that have an output greater than zero will be activated (Sharma et al., 2020). Equation 10 defines the ReLU activation function (Cuevas et al., 2020).

$$\sigma = \max\,(0, z) \tag{10}$$

3.11.1 Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) is a good choice when working with sequential data due to the RNN's ability to retain calculated results from a previous state and apply this data to the current state (Deepak & Chitturi, 2020). RNNs are the backbone of prominent applications like Google's voice search and Apple's Siri (Mittal & Umesh, 2020). RNNs have several advantages over traditional neural networks. Firstly, traditional neural networks usually accept a fixed-size vector as the input and return a fixed-size vector as the output. Secondly, some traditional neural networks use a fixed number of steps (Alom, Moody, Maruyama, Essen, & Taha, 2018). An RNN uses the outputs as inputs for the hidden layers. Though RNNs show major promise, one downside to the architecture is the high computational cost in terms of training time and memory usage (Mittal & Umesh, 2020). Alom, Moody, Maruyama, Van Essen, & Taha (2018) define basic formulae for an RNN, using equations 11 and 12:

$$h_t = \sigma_h(w_h x_t + u_h h_{t-1} + b_h) \tag{11}$$

$$y_t = \sigma_y(w_y\, h_t + b_y) \tag{12}$$

Where:

- x_t = input vector
- h_t = hidden layer vector
- y_t = output vector
- b_h = bias vector
- w and u being the weights.

Two popular RNN architectures are the Long Short-Term Memory (LSTM) units and Gated Recurrent Unit (GRU).

3.11.1.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory, initially proposed by Hochreiter (1997) was designed to solve the exploding or vanishing gradient problem (Hochreiter, 1997). Long Short-Term Memory (LSTM) architecture consists of memory units that contain an input gate, output gate, and forget gate (Sak, Senior, & Beaufays, 2014). The input gate controls the flow of information into the memory cell; the output gate controls the output of information from the memory cell into the neural network, whilst the forget gate scales the current state of the memory cell before adding data from a previous state (Sak et al., 2014). From a high-level view, the LTSM memory cell can be illustrated as follows:

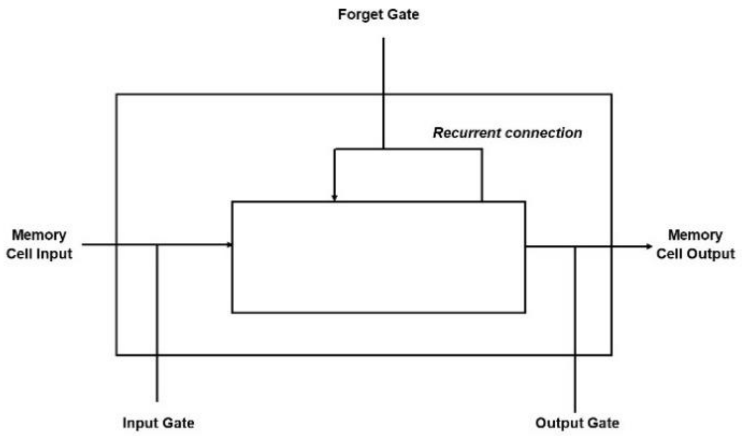


Figure 3.13: High-level view of the LSTM Memory cell (Bahad, Saxena, & Kamal, 2019).

3.11.1.2 Gated Recurrent Unit (GRU)

Another variant of the recurrent neural network architecture utilises the Gated Recurrent Unit (GRU). This architecture is said to be simpler and exhibit a lower computational cost (Alom et al., 2018). Similar to the LSTM cell, the GRU has gates that control the flow of information; the difference is that the gates are not contained in separate cells (Chung, Gulcehre, Cho, & Bengio, 2014). The GRU's update gate combines the input and output gates, and the cell's state and hidden state are merged. The update gate controls the extent to which a GRU cell should update the activation function or content. Owing to the GRU harbouring a simpler design, the resultant training time is lower than the LSTM's training time (Mittal & Umesh, 2020).

The output, update and reset functions of the Gated Recurrent Unit (GRU) cell can be represented, using equations 13, 14, 15 and 16 (Chung, Gulcehre, Cho, & Bengio, 2014).

$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j \hat{\mathrm{h}}_t^j \tag{13}$$

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^t \tag{14}$$

$$\hat{\mathrm{h}}_t^j = \tanh\,(W x_t + U(r_t \odot h_{t-1}))^j \tag{15}$$

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \tag{16}$$

Where:

- h_t^j = activation of a GRU cell at time *t*
- z_t^j = update function for update gate
- h_{t-1}^j = previous activation function for GRU cell
- $\hat{\mathrm{h}}_t^j$ = next activation function
- r_t^j = reset function

3.11.2 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) embody a simple structure and are effective in extracting features from the text in an environment where computational resources are limited (Li, Li, & Lo, 2020). The most common use-cases for Convolutional Neural Networks are image classification tasks (Phung & Rhee, 2019). At each convolution layer, a convolution filter slides through every part of the input, producing a resultant feature map. Every part of the input is multiplied with the filter and then used in an applicable activation function, which in turn produces a feature map for the convolutional filter (Li, Li, & Lo, 2020). Following the creation of a resultant feature-map, is the process of max-pooling, which selects the most relevant feature (highest value) from a feature map for a filter (Kim, 2014). Figure 3.9 presents a high-level representation of a Convolutional Neural Network.

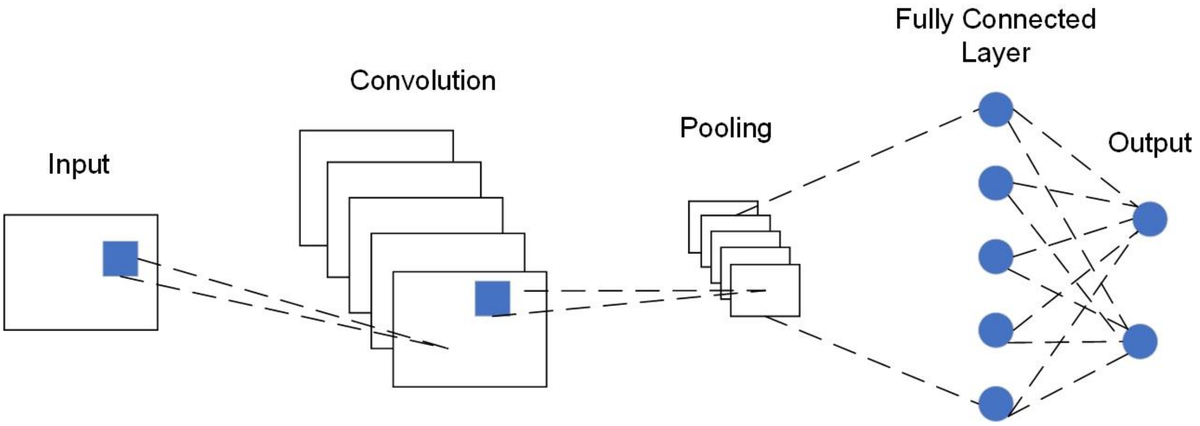


Figure 3.14: Anatomy of a typical Convolutional Neural Network (Phung & Rhee, 2019)

3.11.3 Hybrid Deep Learning Models

Researchers in online misinformation detection have explored and demonstrated the effectiveness of hybrid deep learning models in the detection of fake news. Hybrid deep learning models are a combination of multiple deep learning architectures. Nasir, Khan, & Varlamis (2021) propose a fake news detection model by combining two artificial neural network architectures – the Convolutional Neural Network and Recurrent Neural Network. The premise of introducing such a model was to address the limited research in fake news detection, where a combination of multiple deep learning architectures are used (Nasir et al., 2021). The authors use the Convolutional Neural Network to extract latent features from the text, and the Long-Short Term Memory (LSTM) cell for the Recurrent Neural Network, to extract long-term dependency related features from the text (Nasir et al., 2021). Using the hybrid model, the authors attain their highest accuracy reading, at 99%, using the ISOT dataset (Nasir et al., 2021).

3.11.4 Related Work: Deep Learning and Fake News Detection

Kaliyar, Goswami, & Narang (2020) explore the problem of fake news detection through the use of a machine learning approach, and a deep learning approach. The authors use BuzzFeed (Mahudeswaran & Shu, 2019) and PolitiFact datasets for the experiments. In the machine learning approach, the authors use the XGBoost machine learning algorithm in three separate experiments. The first experiment considers features related to the contents of articles contained in the news dataset, the second considers features related to user interactions for articles, and the third experiment considers the news-user interactions (Kaliyar, Goswami, & Narang, 2020). An n-gram feature set is employed to represent the contents of articles in the dataset. In a deep learning approach, the authors create DeepFake, a neural network consisting of four hidden layers (Kaliyar et al., 2020). In the neural network, the first, second and third hidden layers consist of 128 neurons whilst the last layer consists of 32 neurons (Kaliyar et al., 2020). The DeepFake model attains a precision of 82.10%, recall of 84.60% and F1-Score of 84.04%. Using the BuzzFeed dataset, the model attains a precision of 83.33%, recall of 86.96% and an F1-Score of 85.11% (Kaliyar et al., 2020).

Goldani, Safabakhsh, & Momtazi (2021) explore the use of deep learning technology using a Convolutional Neural Network with margin loss as the cost function and word2vec-generated word embeddings. The authors use the LIAR and ISOT datasets in the experiment. In the classification experiments, the authors show that their CNN model with margin-loss as the cost function outperforms a CNN with cross-entropy as its cost function by 2.1% (Goldani et al., 2021).

Deepak & Chitturi (2020) present an approach that explores the problem of online fake news detection through neural network and data mining techniques. The authors argue that the detection of online fake news can be addressed through the creation of a feature set that considers the contents of articles, as well as features that describe the interactions of users on articles. In the experiment, the authors employ two deep learning models: the Long Short Term Memory (LTSM) architecture and Feed Forward architecture (Deepak & Chitturi, 2020). GloVe, word2vec and bag of words (BoW) models were chosen, creating vectors that represent articles in the respective datasets. Using features obtained from data mining techniques, the authors achieve as much as an 8% increase in model performance versus the same model without data mining-related features (Deepak & Chitturi, 2020). Using the LSTM architecture and data mined features, the authors attain 91,32 % accuracy, 89,19% precision, 94.21% recall and an F1 score of 91,63%. Excluding features obtained through data mining, the authors obtain an accuracy of 83,66%, precision of 79,03%, recall of 92,02% and an F1 score of 85,03% (Deepak & Chitturi, 2020).

In a social media context, Ranjan & Gupta (2021) design a framework that addresses online fake news detection by examining Facebook users' account data alongside the news article through the use of machine learning and deep learning technologies. The framework includes the development of a Google Chrome plugin that displays a popup window on the user's browser. A web crawler and Facebook API is used to collect information relating to the user and the article (Ranjan & Gupta, 2021). The machine learning classifiers selected include the Support Vector Machine, Logistic Regression, Decision Tree, and Naïve Bayes, whilst the deep learning architecture selected is the Long Short Term Memory (LSTM) (Sahoo & Gupta, 2021). Of all the machine learning algorithms, the authors found the Support Vector Machine provided a better accuracy score compared to other classifiers. Additionally, it was found that the deep learning model yielded a better accuracy score than the machine learning classifiers. In the experiment, the LSTM architecture, considering user profile features and news features, achieve a high accuracy score of 99.4%, thus outperforming machine learning classifiers using the same feature set (Sahoo & Gupta, 2021).

In an alternative view, Zhang, Dong, & Yu (2016) describe the problem of fake news detection as one which can be solved by observing the credibility of news articles and sources. The intuition behind such a view is that credible sources will have a high credibility score while less credible sources will have a low credibility score. The authors construct the FakeDetector framework, which aims to determine the credibility of articles, authors and article subjects by examining the relationships between such entities (Zhang, Dong, & Yu, 2016). To extract hidden (latent) features from the union of articles, news subjects and authors, the Recurrent Neural Network (RNN) with a Gated Recurrent Unit (GRU) architecture is employed. The authors also introduce the Hybrid Feature Learning Unit (HFLU), which can learn hidden features from the explicit feature set. The network consists of three layers; one input layer, one hidden layer and one fusion layer (Zhang, Dong, & Yu, 2016). Whilst extracting explicit features, the authors construct a vocabulary of all unique words in the PolitiFact datasets, and subsets of unique words that appear in articles, subjects and authors (J. Zhang et al., 2016). The resulting feature sets for articles, subjects and authors represent words and the number of times a said word appeared in an article (Zhang, Dong, & Yu, 2016). The explicit features and latent features are concatenated and fed into a deep diffusive network.

Some researchers have explored the combination of various artificial neural network models in the fake news detection task. Drif, Hamida, & Giordano (2019) explore fake news detection by creating a CNN-LSTM neural network that combines the Convolutional Neural Network and Long-Short Term Memory architectures. The first layer in the network, the embedding layer, accepts vectorized textual data as input, in the form of a vector space. A pre-trained, 300-dimensional GloVe model using Google News vectors was used in generating word embeddings (Drif, Hamida, & Giordano, 2019). Following the embedding layer, a drop-out layer was added to prevent over-fitting. A CNN layer, with 10 filters of size 3, and a ReLU activation function, takes input from the previous dropout layer (Drif, Hamida, & Giordano, 2019). A max-pooling layer is selected to reduce the size of the input by selecting relevant (highest) features. Following the max-pooling layer, an LSTM layer is employed and accepts inputs from the max-pooling layer. The last layer, being the dense layer, uses the sigmoid function and converts the array into a range. The LIAR (Wang, 2017) and News Articles datasets were used. Through experimentation, Drif, Hamida, & Giordano (2019) found the CNN-LSTM model outperformed the Support Vector Machine (SVM) model, CNN, and RNN models, the highest accuracy score being 72.50% (Drif, Hamida, & Giordano, 2019).

Similarly, Agarwal, Mittal, & Goyal (2020) use a model for fake news detection that combes RNN (LSTM) and CNN architectures. Using a publicly available Kaggle dataset (Kaggle, 2018) from a Kaggle competition, the authors consider a given article's title and text in the binary classification of fake news (reliable, unreliable). The authors note that RNNs come with high computational costs when working with large datasets; a solution to the problem could involve the use of a CNN to extract features from the data (Agarwal, Mittal, & Goyal, 2020). From a high-level view, the proposed model uses two 1-Dimensional convolutional layers, a max-pooling layer, and an LSTM layer. Using a model which combines CNN and LSTM architectures, the authors achieve a precision score of 97,21%, a sensitivity score of 91,89%, a specificity score of 91,89% and a training accuracy of 99.54% in the model's last epoch (Agarwal, Mittal, & Goyal, 2020). The GLoVe model was used in generating word embeddings for articles in the dataset.

Using the Convolutional Neural Network (CNN) architecture, Kaliyah, Goswami, Narang, & Sinha (2020) develop the FNDNet model. To validate the efficacy of the FNDNet model, the authors select 4 machine learning algorithms and 2 other deep learning architectures. To generate word embeddings from the text, the authors select pre-trained GloVe and Word2Vec models for the experiment (Kaliyah, Goswami, Narang, & Sinha, 2020). The 4 selected machine learning models are Multinomial Naïve Bayes, Decision Tree, Random Forest, and K-Nearest Neighbour. The 2 selected deep learning architectures were Convolutional Neural Network, Recurrent Neural Network (Long Short Term Memory) (Kaliyah, Goswami, Narang, & Sinha, 2020). In the experiments, the authors find the FNDNet model, using GloVe embeddings, delivered superior results, compared to the other models, with an accuracy of 98,36%, a precision score of 99,40%, a recall score of 96,88% and an F1 score of 98,12% (Kaliyah, Goswami, Narang, & Sinha, 2020).

In a deep learning context, Choudhary & Arora (2020) devise a sequential neural network model for the detection of online fake news. Syntactical features, sentiment features and grammatical features are provided as inputs to the neural network. Through the process of fine-tuning, the authors construct a deep learning network that consists of two hidden layers, with each layer consisting of four neurons (Choudhary & Arora, 2020). At the two hidden layers, the ReLU activation function is selected, whilst at the output layer, the SoftMax function is employed (Choudhary & Arora, 2020). The authors define the fake news detection problem as a binary classification problem; an article is either truthful or fake. The feature groups are split into three models. Model 1 consists of syntactical, sentimental, and grammatical features; model 2 consists of readability features; model 3 is a combination of models 1 and 2 (Choudhary & Arora, 2020). Highlights from the experiment show that model 1 outperforms models 2 and 3, achieving an accuracy rate of 82% (Choudhary & Arora, 2020). Models 2 and 3 attained accuracy rates of 72% and 80.22% respectively.

Numerous researchers propose models for the detection of online fake news, which evaluate both the textual contents and visual contents (images). Singhal, Shah, Chakraborty, Kumaraguru, & Satoh (2019) propose the SpotFake – a fake news detection framework that considers both text and images for the task of fake news detection. SpotFake consists of three modules. The first module uses Bidirectional Encoder Representations from Transformers (BERT), a language representation module, to extract textual features from the data. The second module employs a pre-trained VGG19 convolutional neural network to extract image features. Finally, the last module combines textual features and image features and feeds the combined vectors to a fully connected neural network for classifying articles as fake or real news (Singhal et al., 2019). BERT is a bidirectional language-based model, proposed by Kenton, Kristina, & Devlin (2019) which has shown promising results for several natural language processing tasks (Devlin, Chang, Lee, & Toutanova, 2019).

Kula & Chora (2020) explore the use of deep learning technology, specifically recurrent neural networks with the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) architectures, using the Flair library. At the time of publication, the authors noted that the implementation of deep learning models in fake news detection using the Flair library had not been covered (Kula & Chora, 2020). The ISOT Fake News Dataset and Getting Real about Fake News (GRaFN) are selected to train two separate deep learning models. Using GloVe word embeddings, the authors achieve the highest accuracy, recall and precision scores of 99.86%, 99.81% and 99.82% respectively (Kula & Chora, 2020). Owing to the definition of fake news being wide (inclusive of one or more subcategories like propaganda or conspiracy), the true intent behind an article may not always be clear by examining just the article's textual data (Kula & Chora, 2020).

3.12 Machine Learning and COVID19

With COVID19 misinformation on the rise, researchers have explored various machine learning and deep learning solutions which could be used in the fight against COVID19 misinformation. Abdelminaam, et al. (2021) explore automated detection of COVID19 online misinformation, using six machine learning algorithms, namely, Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbour (KNN), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), and two recurrent neural network architectures, such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The authors use four Twitter datasets that span a range of topics, such as healthcare misinformation, politics, disaster management, and general gossip (Abdelminaam, et al., 2021). In their findings, the authors note the LSTM model achieved the best cross-validation results compared to the six observed machine learning models (Abdelminaam, et al., 2021).

3.13 Conclusion

Machine learning and deep learning technologies have shown great promise in combatting online misinformation. The continuous development of datasets that can be used for research into online fake news further justifies the relevance of the work done in this research area. The literature mentions several machine learning and deep learning models used by several researchers, together with the benefits and disadvantages of using some models for the problem.

This chapter aims to provide extensive coverage of machine learning technology and how the technology has been applied to combat online misinformation. Chapter 4 provides an in-depth view of the research process and design for the study.

4. Chapter 4 - Research Methodology and Design

This section presents the research methodology and design undertaken in this research project. As mentioned in Chapter 1, the objective of this study is to design a set of guidelines that could be used in developing tools or solutions for automated fake news detection.

4.1 Overview

The purpose of devising a research methodology and design plan is to outline a systematic process as to how knowledge will be uncovered. A research methodology describes a systematic way of undertaking research and solving research questions (Sarkar & Sahu, 2018). Quantitative research methods collect numeric, statistical, measurable data to provide answers for a phenomenon (Earl, 2010). In contrast, qualitative research involves the collection of data through non-numeric means such as using questionnaires, observations, and audio.

A research project can take a positivism or interpretivism approach. Positivism research bases findings on factual and logical information (Hughes, Sharrock, Hughes, & Sharrock, 2011). In positivism, data is presented in its raw form, without further processing and interpretation (Hughes et al., 2011). Positivism often includes experimentation and quantitative research. Positivists typically formulate conclusions and findings on facts that can be verified (Ryan, 2018). In contrast, interpretivism deals with the central theme of truth being subjective, depending on several factors.

4.2 Research Design

The research takes a positivist approach to uncover facts and information. Quantitative research methods are employed in the research process. Through a series of experiments, constructed in a systematic manner, knowledge and facts related to the applicability of automated fake news detection are uncovered. Furthermore, the results and data collected during the experimentation phase are discussed and compared against findings by other researchers, through Logical Argumentation.

4.2.1 Literature Review

The purpose of the literature review is to better understand the research problem and examine existing literature related to the research problem. The literature review extensively covers the idea of fake news, why it is a problem and why such a problem is worthy of investigation. Through this method, the literature review reveals possible indicators which may suggest a given article may be fake news. These indicators are crucial for the experimentation phase, where features that best fit such indicators are selected. In addition, the literature review makes clear numerous machine learning and deep learning models which have been applied by researchers, in the task of misinformation detection. The advantages and disadvantages of such models determine which models are selected for experimentation. The literature review addresses secondary research objectives 1 and 2:

1. *Identify the motives behind the propagation of fake news.*
2. *Identify appropriate indicators, from literature, which could suggest an article is fake news.*

4.2.2 Experimentation

Through experimentation, the notion of detecting online fake news is tested using technologies highlighted in the literature review, namely Machine Learning and Deep Learning technology. The purpose of the experiment is to validate the effectiveness of fake news detection, using a collection of features derived from identified indicators. To determine the effectiveness of the experiments, a range of well-known performance metrics are selected. The objective of the experiment is to provide answers to research secondary objective 3:

3. *Validate the identified indicators through the application of selected machine learning algorithms.*

The experiments are set up systematically, like other researchers who have worked in this field of study. A discussion and a brief analysis of the results are given in this chapter.

4.2.3 Logical Argumentation

Finally, logical argumentation is employed to examine the results obtained in the experimentation chapter. Argumentation aims at convincing critics of the acceptability of your viewpoint, using a collection of statements as the premise (Van Eemeren & Grootendorst, 2004). Logical argumentation is used to discuss the results obtained, compare them with results that other researchers have obtained, and position the results alongside other research into fake news detection using technological solutions. Logical argumentation is employed as a means to systematically set up experiments related to this project, based on experimental set-ups used by other researchers in the fake news detection community.

Using information collected through literature review, data collected through experimentation, and findings discussed through logical argumentation, guidelines into fake news detection can be formulated. Figure 1.1 illustrates the overall research design process undertaken for the project.

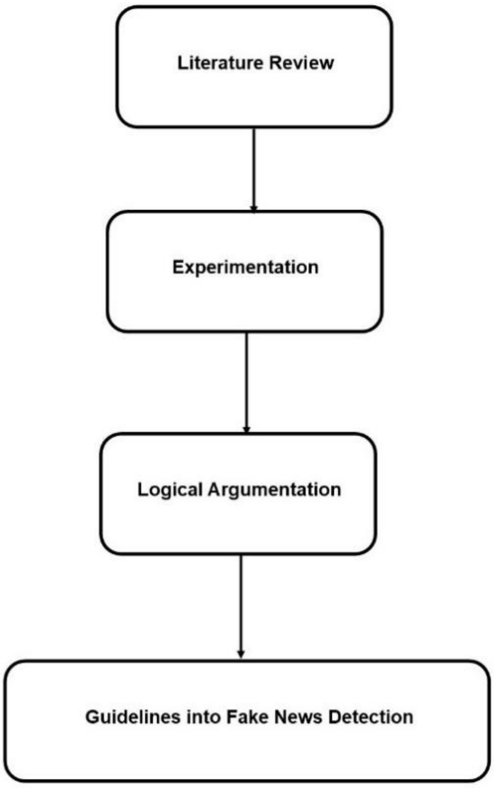


Figure 4.1: Overall research process employed in this study

4.3 Experiment Implementation Details

To carry out the experimentation, a number of tools, packages and benchmarking standards are required. The Python programming language is selected for the construction of the selected machine learning and deep learning models. The PyCharm IDE is selected for creating the necessary Python scripts. To construct the necessary machine learning and deep learning models, the Sci-Kit Learn and Keras Python libraries are selected respectively. Finally, to determine optimal configuration for the deep learning models, the Keras-Tuner library is selected.

To quantify the performance of the machine learning and deep learning models, performance metrics such as accuracy, precision, recall, and F1-score are selected.

These metrics derive their results from values present on the confusion matrix, such as the number of true positives, false positives, false negatives and true negatives. The formulae for the selected performance metrics are discussed in Chapter 3. Further details surrounding the use of machine learning and deep learning packages are discussed in Chapter 5 and Chapter 6.

4.4 Guidelines For the Detection of Text-Based Fake News

Using data collected through experimentation, and the interpretation of the results through logical argumentation, the deliverable for this project, *guidelines for the detection of text-based fake news* are constructed. The purpose of the guidelines is to list key points which should be considered when developing systems for text-based fake news detection. The practicality of such guidelines is supported by findings in the experimentation phase and findings from other authors in the literature review.

4.5 Conclusion

This chapter presents a detailed overview of the research process employed in the project. This chapter also makes it clear how the selected research methods address the primary and secondary objectives and contribute towards the development of machine learning guidelines for text-based fake news detection. Chapter 5 covers the experimentation phase of the project.

5 Chapter 5 - Experiment

This chapter presents the experiment undertaken in the research project. The objective of the experiment is to determine the effectiveness of automated tooling in the detection of fake news, through examining various combinations of identifiers and machine learning algorithms.

5.1 Introduction

The experiments covered in this chapter aim to determine the effectiveness of machine learning and deep learning approaches for the task of automated fake news detection. The task of detecting fake news is a complex task for a myriad of reasons, such as language, text lengths, and additional metadata. In Chapters 2 and 3, indicators that could suggest a given article or statement is fake are outlined; such indicators are discussed in greater lengths in this chapter.

The experiment explores and provides an answer to secondary objective 3 of the research objectives:

3. *Validate the identified indicators through the application of selected machine learning algorithms.*

In Chapter 2, the possible indicators of a given article being a fake news article are outlined. Given the findings outlined by various researchers, the following properties of online news articles are considered as indicators that could differentiate a fake news article from a real news article:

1. *Fake news articles generally contain short article bodies and longer article titles.*
2. *Fake news articles generally use simpler sentence structures.*
3. *Fake news article's generally have headlines which aren't related to the article body.*

In the series of experiments that follow, a broader definition of fake news is applied; that is, any articles that mislead online readers by publishing unverified facts or false content. As such, news articles can be fit into one of two classes: truthful or false.

5.2 Hardware Configuration

The experimentation undertaken in this study involves the use of machine learning and deep learning approaches to text-based fake news detection. The experiments are performed offline, on a notebook computer. The notebook has a 500GB NVMe M.2 SSD, an Intel Core i5-7700HQ processor, 16GB of DDR4 RAM, and an Nvidia GeForce GTX 1050 4GB Graphics card, which is useful for deep learning. The notebook uses Windows 10 Pro (64 bit) as the operating system. All Python scripts are written and executed using the JetBrains PyCharm IDE.

5.3 Datasets and Tooling

Figure 5.1 illustrates the overall process employed to generate a feature set which best describes the problem. In the experiment, two datasets are used. Dataset A is a collection of the online news articles used for the classification task, whilst dataset B is used to train the Doc2Vec model to generate document-based vectors. Following data pre-processing, feature extraction processes ensue. Finally, the generated document vectors are concatenated with the generated features, creating an initial feature set for the experiment.

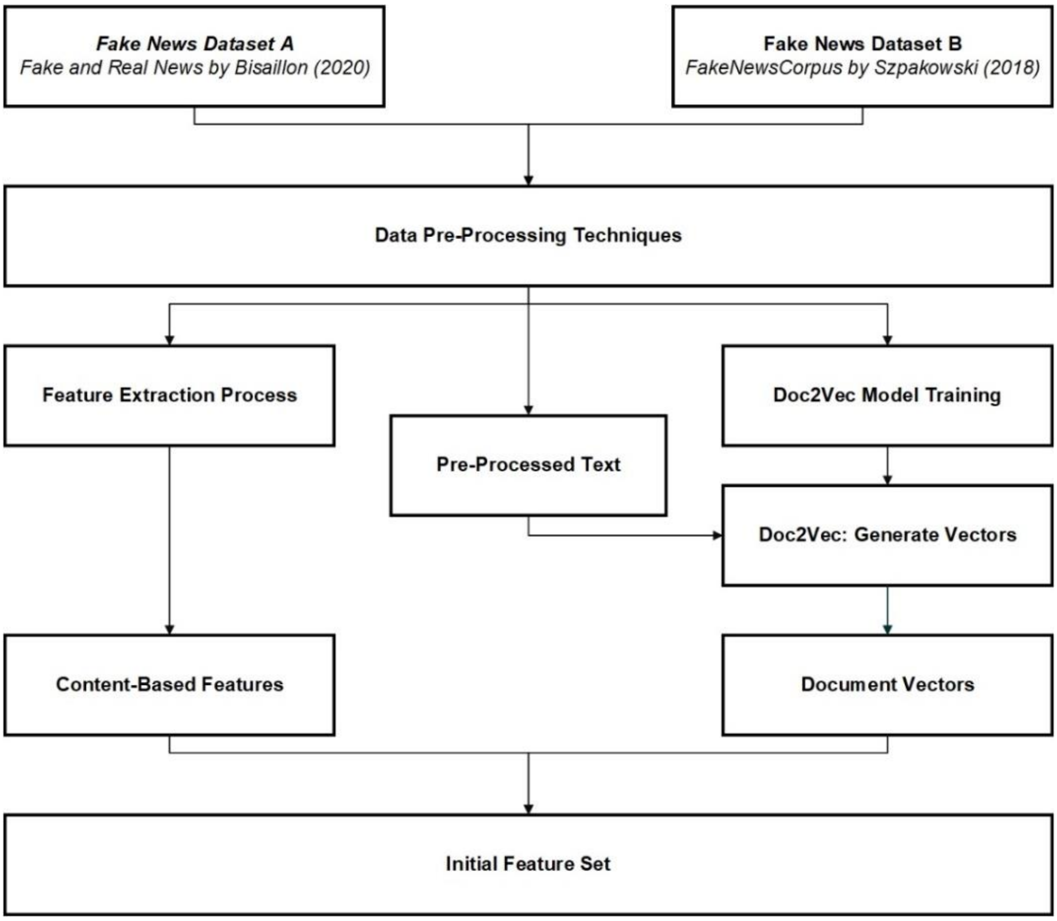

Fake News Dataset A
Fake and Real News by Bisaillon (2020)
Fake News Dataset B
FakeNewsCorpus by Szpakowski (2018)
Data Pre-Processing Techniques
Feature Extraction Process
Pre-Processed Text
Doc2Vec Model Training
Doc2Vec: Generate Vectors
Content-Based Features
Document Vectors
Initial Feature Set

Figure 5.1: Overall feature extraction process employed in the experiment.

5.3.1 Datasets

Two datasets are employed in the experiments. One dataset is used for the classification task of discerning between false and real news articles, and the second dataset is used to train a doc2vec model to generate document vectors for each article contained in the first dataset. The fake-real news dataset created by Bisaillon (2020) is used for the classification task, and the FakeNewsCorpus dataset, created by Szpakowski (2018) is used to train a doc2vec model. The Bisaillon (2020) dataset contains 23 481 fake online news articles and 21 417 real online news articles published between the years 2015 and 2018. Real news articles were collected from Reuters.com, an online news firm. Fake news articles were collected from websites that were flagged as 'unreliable' by PolitiFact (Bisaillon, 2020). The fake news articles span six categories, namely, government news, Middle-East news, US news, left-news, politics, and news (Bisaillon, 2020)

The FakeNewsCorpus dataset, created by Szpakowski (2018), contains over 9.5 million articles and is 28 GB in size. Owing to the limited memory and processing power present on the computer used to perform the experiments, the first 12 000 articles, spanning over all news categories contained in the dataset, are selected. The merged dataset is a collection of 120 000 articles, evenly distributed across 10 news categories. Table 5.1 shows all news article categories and the number of articles selected for each category.

Table 5.1: Summary of news article categories for selected articles used to train the Doc2Vec model

Article Category	**Number of Selected Articles**
Fake News	12 000
Satire	12 000
Bias	12 000
Conspiracy	12 000
Junk Science	12 000
Hate News	12 000
Clickbait	12 000
Unreliable	12 000
Political	12 000
Credible	12 000

The diagrams below represent common words associated with fake and real news articles in the dataset. Figure 5.2 and Figure 5.3 represent the topmost 3000 words found in the online fake news and online real news datasets authored by Bisaillon (2020). The most frequent words appear larger than the least frequent words. The SciKit-Learn's TfIdfVectorizer class was used to calculate a TF-IDF score as a way of calculating the significance of words contained in both datasets. The fake news dataset resulted in a vocabulary size of 94137 words, whilst the real news dataset resulted in a vocabulary size of 66 355 words. Using both fake and real news vocabulary sets, a total of 38 806 words were found which were present in both fake and real news vocabularies.

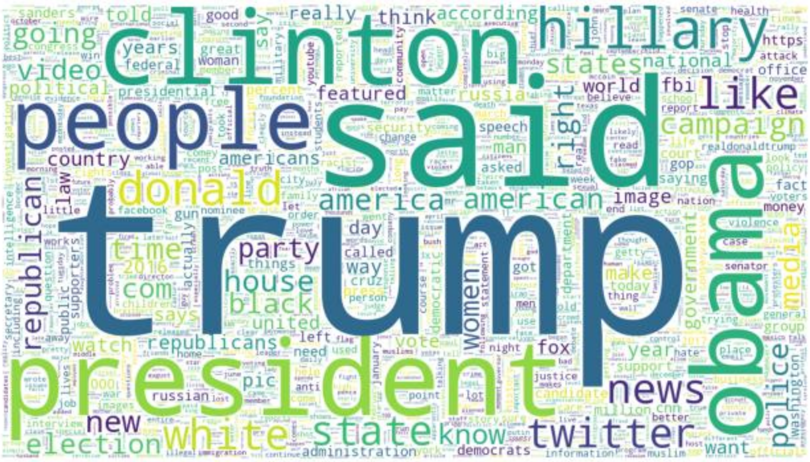


Figure 5.2: Word cloud representation of the most frequent words contained in the online fake news articles dataset, authored by Bisaillon (2020).

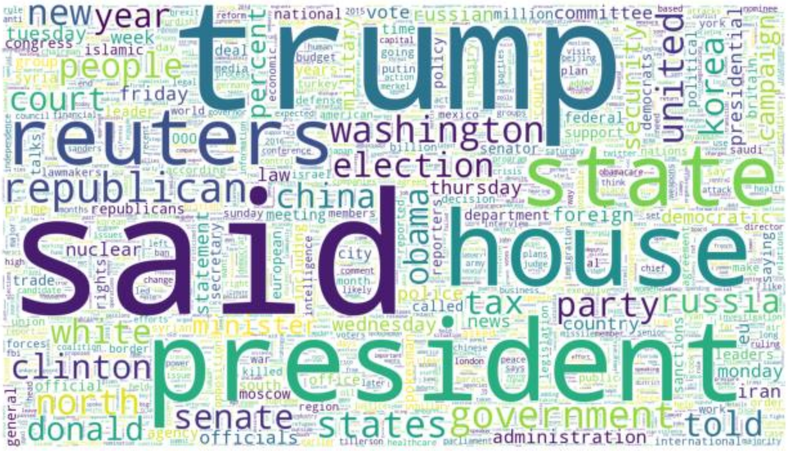


Figure 5.3: Word cloud representation of the most frequent words contained in the online real news articles dataset, authored by Bisaillon (2020)

5.4 Data Pre-processing

This research employs various data pre-processing techniques to ensure the raw, textual article data is in a state ready for the feature extraction and selection processes. This step of the process mostly involves the removal of noise in the data.

5.4.1 Stop words and special character removal.

Stop words removal is a process where common words, which add little value to text, are removed from bodies of text. Special characters and any other common words, such as is, as, etc. are removed from articles.

5.5 Feature Extraction

Having considered the information presented in the literature, it is evident that a tool for fake news detection should consider the article title, article contents, the relationship between the title and body, and the article's anatomy. In this study, the work of Masood & Aker (2018) is used as a reference for possible features that capture the points above. Such features are listed below:

5.5.1 Title and Article Body Word Count

Total counts on the number of words contained in article titles and article bodies are kept as separate metrics for each article. The inclusion of this metric as a feature is based on an observation by Horne & Adali (2017), who state fake news articles have shorter article bodies.

5.5.2 Title and Body Punctuation Count

The total number of punctuation marks found in an article's body is added to the feature matrix. The rationale supporting the inclusion of this metric in the feature set is based on several observations relating to title length, sentence length and punctuation counts in fake and real news articles.

5.5.3 Sentence Length

For each article, the average sentence length for each article in the corpora is calculated. The inclusion of this metric relates to the observation that fake news articles generally contain shorter article bodies and sentences, compared to truthful articles.

5.5.4 Type-Token Ratio

The type-token is calculated by tallying the total number of unique words in a document and dividing this by the total word count present in a document. To calculate the type-token ratio for article bodies contained in the dataset, the LexicalRichness (YS, 2018) Python module is used. The type-token ratio can be calculated using equation 17 (YS, 2018).

$$tt\_ratio = \frac{Nt}{Nw} \tag{17}$$

Where:

- Nt: unique words present in each document
- Nw: total word count per document

5.5.5 Cosine Similarity

An averaged cosine similarity score is calculated for each article contained in the dataset. The average score is calculated by considering the article title against all sentences in each article. The inclusion of cosine similarity as a metric to measure the relationship between article titles and texts are supported by the work of Masood & Aker (2018), who employ the metric as a means of measuring the similarity of article titles and article bodies. Given document vectors A and B, cosine similarity is calculated using equation 18 (Han, Kamber, & Pei, 2012):

$$\cos\_sim = sim(A, B) = \frac{A\ \cdot B}{||A||||B||} \tag{18}$$

5.5.6 Text Readability Metrics

Several text readability metrics are selected to measure the readability of article texts. The inclusion of such metrics is based on the work of Horne & Adali (2017), who use text readability metrics as a means of examining article structure. The py-readability-metrics (DiMascio, 2019) Python package is used to perform the necessary calculations. The selected readability metrics are defined below.

5.5.6.1 Gunning Fog Index

This metric derives its score from the US schooling system, indicating the educational grade required to comprehend a piece of text (Bilal & Huang, 2019) The py-readability-metrics package calculates the Gunning Fog Index metric using equation 19 (DiMascio, 2019):

$$text\_gf = 0.4 \cdot (\frac{Tw}{Ts} + 100 \cdot \frac{Cw}{Tw}) \tag{19}$$

Where:

- *Tw*: Total words
- *Ts*: Total Sentences
- *Cw*: Total complex words

5.5.6.2 Flesch Kincaid Grade Level Index

This is another text readability metric where scores correlate to the educational grade required to comprehend a given piece of text (Bilal & Huang, 2019). The py-readability-metrics calculates the related reading index using equation 20 (DiMascio, 2019):

$$text\_fkg = (0.39 \cdot Aws + 11.8 \cdot As) - 15.59 \tag{20}$$

Where:

- *Aws*: The average word count per sentence, per document
- *As*: Average syllables per word, per document

5.5.6.3 Automated Readability Index

This metric considers sentences and the word count. The score relates to the educational grade level required to comprehend a piece of text (Bilal & Huang, 2019). The py-readability-metrics package calculates the related index using equation 21 (DiMascio, 2019):

$$text\_ari = 4.71 \cdot LPW + 0.5 \cdot Ws - 21.43 \tag{21}$$

Where:

- *LPW*: Number of letters per word, per document
- *Ws*: Number of words per sentence, per document

5.5.7 Doc2vec

The doc2vec model is used to transform article texts into document vectors. The model is pre-trained using a subset of articles contained in the FakeNewsCorpus authored by Szpakowski (2018). After model training, article texts from the fake news dataset used for the classification task are fed into this model, which in turn, generates the necessary document vectors.

In the experiment, the Gensim implementation of the Doc2vec model is selected. For the model's configuration, the window size is set to 8, and the Paragraph Vector-Distributed Bag of Words (PV-DBOW) model is selected as the training algorithm. Due to limited computational resources, the first 120 000 articles, spanning over 10 news categories, is selected from the FakeNewsCorpus authored by Szpakowski (2018). The decision to create a subset of articles, containing 120 000 news articles is inspired by Le & Mikolov (2014) sentiment analysis experiment. In the experiment, the selected dataset contained 100 000 movie reviews. In all experiments undertaken, Le & Mikolov (2014) set the window size to 8 and the number of vector dimensions to 400, for both PV-DBOW and PV-DM models (Le & Mikolov, 2014).

5.5.8 Text Summarization

Due to high computational costs associated with training recurrent neural networks, and limited computational resources, a separate dataset consisting of summarized articles are created. Following basic text pre-processing highlighted in section 5.4, the text is sent to the T5 model for text summarization. The experiment uses the Hugging Face (Wolf, et al., 2019) implementation of Google's T5 (Text-To-Text Transfer Transformer) model. As part of the T5 framework release, Google provided several pre-trained models along with source code. This experiment uses the 't5-base' pre-trained model, which contains 220 million parameters. Following summarization, the longest summarized document contained 98 words. Articles containing no text were removed. As a result, 631 rows were removed.

Following basic data cleansing techniques, the first step of the process is to encode the text. This experiment uses the T5Tokenizer to handle this aspect. To summarize text, the input text is pre-appended with the 'summarize:' token, as described by Raffel et al. (2019). To generate document summaries, we set the max_length parameter to 200, num_beams to 4, min_length to 20 and no_repeat_ngram_size to 2. The final step of the process is to decode the summaries into human-readable text. For sequence-to-sequence generation, Hugging Face (2020) recommends using the T5ForConditionalGeneration.generate() method (Hugging Face, 2020).

5.5.9 Parts of Speech Tagging (POS)

For each document, the parts of speech found are collected. Each part of speech contains a count on the number of occurrences found in each article. The Parts of

Speech tags assist in better understanding the grammatical composition of fake and real news articles. This accumulative feature addresses part of the 2nd fake news identifier:

2. *Fake news articles generally use simpler sentence structures.*

5.6 Initial Feature Set

Table 5.2 represents the features extracted from fake and real news articles (Ngada & Haskins, 2020). The dataset authored by Bisaillon (2020) was selected. From the articles, a collection of 49 features that capture sentence structure and composition is created. Document vectors are generated using a trained doc2vec model. The doc2vec model is trained with data from the FakeNewsCorpus dataset. Following model training, document vectors are generated – the resultant vectors are appended to the feature set. The resultant feature set contains 458 vectors.

5.7 Machine Learning Algorithms

The Support Vector Machine, Random Forest, AdaBoost, XGBoost, and K-Nearest Neighbour machine learning algorithms are selected for the experiment. The results of each classifier are discussed in section 5.8.

The resultant feature set, generated from processing the fake and real news dataset, authored by Bisaillon (2020), are provided as inputs into the selected machine learning algorithms. 80% of the dataset is selected as training data, whilst the remaining 20% is selected as the test dataset.

5.7.1 Machine Learning Algorithm Configuration

The GridSearchCV hyperparameter selection technique is employed to determine an optimal parameter set, for each selected machine learning model. Silva et al. (2020) use the grid search technique to determine the best configuration for the number of estimators, and regularization parameters for the AdaBoost, Support Vector Machine, Random Forrest and Bagging classifiers (Silva et al., 2020). Wang (2017) employs grid search to determine optimal parameter configurations for the Logistic Regression and Support Vector Machine models. Table 5.3 illustrates all hyperparameters explored in the grid search process. Due to limited computational capacity, grid search is performed over three folds.

Table 5.2: Initial feature set for fake news detection

Calculated Features	**Name**
text_ari	Automated Readability Index
text_gf	Gunning Fog Index
text_fkg	Flesch Kincaid Grade Level Index
tt_ratio	Type-Token Ratio
cos_sin	Cosine Similarity
Accumulative Features	
Count of unique words	Average sentence length
Title Word Count	Article Body Word Count
POS Tag for ‘Adjective’	POS Tag For ‘Noun’
POS Tag For ‘,’	POS Tag for ‘Proper Noun’
POS Tag For ‘Cardinal Digit’	POS Tag For ‘Verb - Present Participle’
POS Tag For ’.’	POS Tag For ‘Determiner’
POS Tag For ‘Verb - Past Participle’	POS Tag for ‘(’
POS Tag For ‘Preposition’	POS Tag for ‘)’
POS Tag for ‘Adverb’	POS Tag for ‘Verb – Single’
POS Tag for ‘Personal Pronoun’	POS Tag for ‘:’
POS Tag for ‘Adjective’	POS Tag for ‘Verb’
POS Tag for ‘Possessive pronoun’	POS Tag for ‘Modal’
POS Tag for ‘3rd person verb’	POS tag for ‘Proper Noun’
POS Tag for ‘ ” ’	POS Tag for ‘Adverb’
POS tag for ‘Proper Noun’	POS Tag for ‘coordinating conjunction’
POS Tag for ‘Pronoun’	POS tag for ‘Possessive pronoun’
POS tag for ‘Whdeterminer’	POS Tag for ‘Existential’
POS Tag for ‘Adjective’	POS Tag for ‘Adverb - Comparative’
POS Tag for ‘Particle’	POS Tag for ‘Symbol’
POS Tag for ‘to’	POS Tag for ‘Foreign word’
POS Tag for ‘Adverb - superlative’	POS Tag for ‘Possessive wh-pronoun’
POS Tag for ‘Predeterminer’	POS tag for ‘List item marker’
Implicit Document Features	
Doc2Vec Features	

Table 5.3: Parameters and values explored through SciKit Learn's GridSearch

Classifier	**Parameter Configuration**	**Best Configuration**	**Accuracy**
K-Nearest Neighbour	**'n_neighbors'**: [2, 3, 4, 5, 6, 7, 8, 9, 10], **'algorithm'**: ['ball_tree', 'kd_tree', 'brute']	**n_neighbors:** 5 **algorithm:** ball_tree	0.908
Support Vector Machine	**'kernel'**: ['linear', 'rbf'], **'C'**: [0.001, 0.01, 0.1, 1, 10, 100, 1000],	**kernel:** linear **C:** 0.1	0.994
Random Forest	**'n_estimators':** [2, 5, 7, 10], **'max_depth'**: [0, 1, 3, 5, 7]	**'n_estimators:':** 10 **'max_depth':** 7	0.971
XGBoost Classifier	**'n_estimators'**: [7, 10, 50, 100], **'max_depth'**: [1, 3, 5, 7]	**'n_estimators:':** 100 **'max_depth':** 5	0.988
AdaBoost Classifier	**'n_estimators'**: [20, 50, 100, 150], **'algorithm'**: ['SAMME', 'SAMME.R']	**'algorithm':** 'SAMME.R', **'n_estimators':** 150	0.992

5.8 Machine Learning Algorithms Classification Results

To quantify a machine learning algorithm's classification performance, five metrics are selected. These metrics are Precision, Accuracy, Recall, F-Measure and Receiving Operator Characteristic(ROC) curve. The confusion matrix models provide data points that the five metrics use. Data normalization and feature selection techniques were not considered. The selected classifiers are configured using optimal configuration as outlined in Table 5.3. The dataset is split into training and testing sets; 80% of data is used as training data, whilst the remaining 20% is used as testing data.

In the experiment, Support Vector Machine is denoted as *SVM,* Decision Tree as *DT,* K-Nearest Neighbour as *KNN,* AdaBoost Classifier as *AB,* and XGBoost Classifier *as XGB.* Table 5.4 shows precision, accuracy, recall, F-Measure and ROC scores attained for each classifier. Table 5.5 shows the accuracy rates obtained at each fold, in 10-fold cross-validation, for each of the selected classifiers. In table 5.6, each numeric column represents the iteration (the fold) in the k-fold cross-validation. Finally, figures 5.4 to figure 5.8 illustrate the area under the curve graphs for the selected machine learning algorithms.

Table 5.4: Performance results obtained through each classifier, following the hyperparameter selection process.

Classifier	Precision	Accuracy	Recall	F- Measure	AUC
Support Vector Machine	0.993	0.994	0.994	0.994	0.993
Decision Tree Classifier	0.950	0.949	0.953	0.952	0.948
Random Forest Classifier	0.994	0.989	0.984	0.989	0.999
K-Nearest Neighbour	0.964	0.912	0.864	0.912	0.915
XGBoost Classifier	0.991	0.988	0.986	0.988	0.988
AdaBoost Classifier	0.993	0.993	0.994	0.993	0.993

Table 5.5: K-Fold cross-validation results obtained with each classifier.

Classifier	1	2	3	4	5	6	7	8	9	10
SVM	0.994	0.994	0.994	0.994	0.994	0.994	0.994	0.994	0.992	0.996
RF	0.988	0.989	0.985	0.990	0.989	0.988	0.988	0.988	0.987	0.989
KNN	0.911	0.913	0.914	0.906	0.920	0.914	0.919	0.916	0.911	0.911
AB	0.990	0.992	0.993	0.991	0.989	0.991	0.992	0.991	0.993	0.992
XGB	0.990	0.988	0.989	0.989	0.985	0.991	0.987	0.989	0.990	0.987

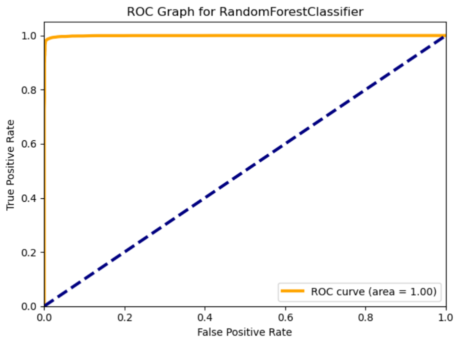


Figure 5.4: ROC AUC for Random Forest

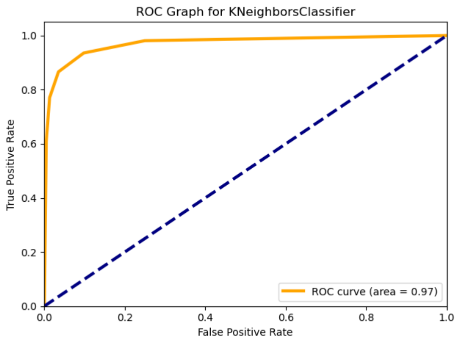


Figure 5.5: ROC AUC for K-Nearest Neighbours

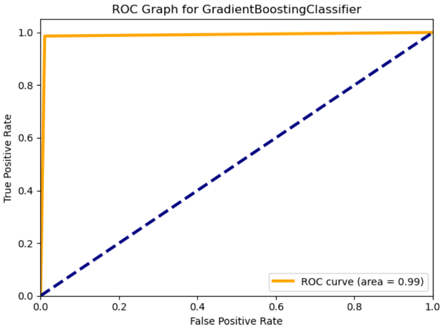


Figure 5.6: ROC AUC for XGBoost

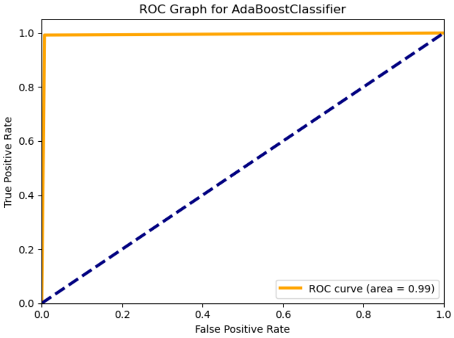


Figure 5.7: ROC AUC for AdaBoost Classifier

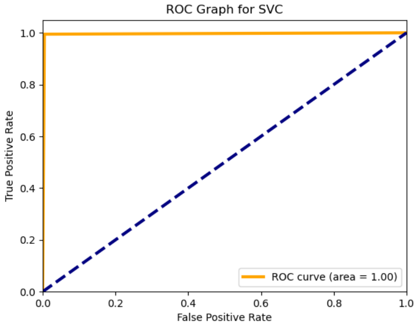


Figure 5.8: ROC AUC for Support Vector Machine

5.9 Deep Learning Experiments

The following section describes the experiments undertaken using deep learning approaches. This part of the experiment uses the same dataset described in section 5.2.1 and the same feature set described in section 5.5. The experiments described in this section were carried out using the Keras (Chollet, et al., 2020) Python library. The Keras-Tuner (O'Malley, et al., 2019) library is selected to perform the hyper-parameter selection process. The experiments were performed on a notebook with an Nvidia GeForce GTX 1050 4GB GPU, Intel Core i5-7300 HQ CPU and 16GB of DDR4 RAM. The operating system present on the notebook is Windows 10 Pro 64-bit. The experiments were done through a series of Python scripts, written using the JetBrains PyCharm IDE (Professional 2020.3).

5.9.1 Deep Learning Architecture Selection and Design

In the exploration of deep learning architecture, several designs, as described by several researchers, are employed. The experiments below use a selection of a hybrid neural network and vanilla neural network designs used by other authors. The Fake

Real News dataset authored by Bisaillon (2020) is selected. A brief explanation of each architecture is described in the sections below.

5.9.2 Hybrid Convolutional Neural Network

In the experiment, fake news detection through the use of a hybrid Convolutional Neural Network that can draw on explicit (handcrafted) features, and implicit features present in the articles is explored. The decision to explore such a design is supported by the work of Yang et al. (2018), who construct a model for fake news detection that uses both explicit and implicit features present in the body of text. The textual features used are described in section 5.5. The set contains 53 features. Following the concatenation of the explicit and implicit text features branches, the following Dense layer is configured to contain 128 neurons, and the output layer is configured to have 1 neuron. For the Dense and Conv1D layers within the model, the ReLU activation function is selected. The output layer is configured to use the sigmoid function as the activation function.

To extract latent features in the articles, the Convolutional Neural Network architecture is employed. The decision to include the CNN architecture is supported by a strategy noted by Agarwal et al. (2020): To overcome the high computation costs exhibited by Recurrent Neural Networks (RNN), the authors use the CNN architecture to extract features from the article text. Furthermore, the combination of explicit and implicit features is used in the work of Yang et al. (2018), who extract features from text and images in both explicit and implicit contexts. This experiment considers only the textual data of articles. Figure 5.9 depicts a high-level view of the proposed hybrid convolutional neural network. For the Dense and Conv1D layers found at numerous layers in the hybrid CNN model, the ReLU activation function was selected. The sigmoid activation function is used at the output layer of the hybrid CNN model.

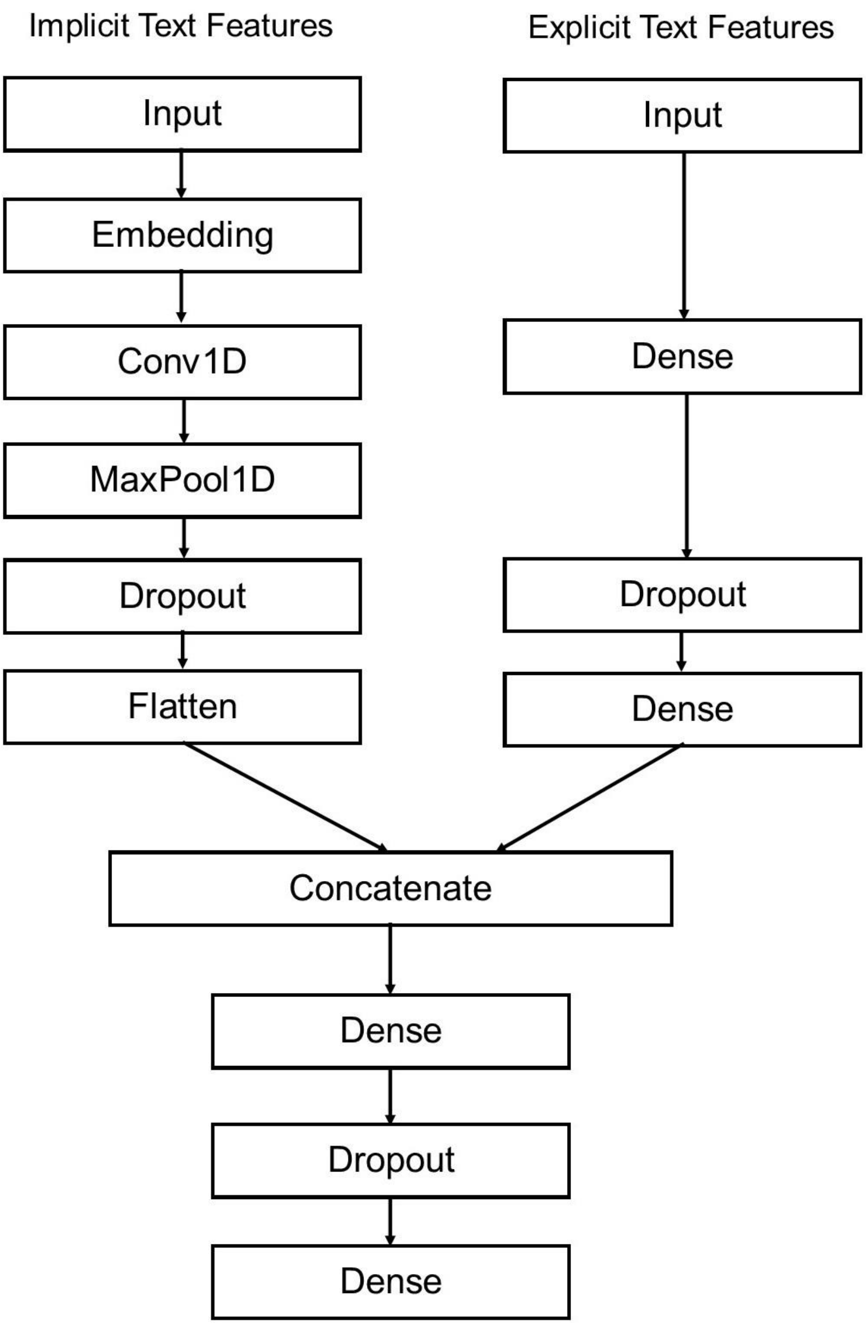


Figure 5.9: Overview of the proposed hybrid CNN-DNN neural network

5.9.2.1 Hyperparameter selection process

To determine the best parameter configuration for the convolutional neural network, the Keras-Tuner (O'Malley, et al., 2019) library is selected. Table 5.7 shows selected parameters explored in the hyperparameter selection process. The number of executions is set to 2 and the number of trials per execution is set to 3. The hyperparameter selection process employed in Keras-Tuner includes creating the model and feeding the model training and testing data. As such, the Fake Real News dataset authored by Bisaillon (2020) is selected. The dataset is split into an 80% training set, and 20% testing set. Basic data pre-processing techniques outlined in section 4 are applied. The hyperparameter selection was done over 20 epochs. Word embeddings for the model were generated using a pre-trained news word2vec model. The resultant word embeddings were used as weights for the Convolutional Neural Network. Table 5.6 shows the final configuration selected for the Convolutional Neural Network. Table 5.7 illustrates the configuration selected at each trial of the hyperparameter process. Due to the model consisting of multiple inputs, layers that are associated with the explicit textual features are signified with *E* whilst layers associated with implicit text features are signified with *I*. Finally, *M* signifies the last 2 layers following concatenation of the two models.

Table 5.6: Parameters selected for Hyperparameter selection process for the Convolutional Neural Network (CNN)

Parameter	Tested Values	Best Values
E: Dense: units	160, 192, 256, 384	384
E: Dropout: rate	0.2, 0.3, 0.4, 0.5	0.5
E: Dense: units	32, 48, 64, 96	48
I: Conv1D: filters	50, 70, 90, 120	90
I: Conv1D: kernels	3, 4, 5	4
I: Conv1D: strides	2, 3, 4	4
I: MaxPool1D: pool_size	3, 4, 5	3
I: Dropout: rate	0.3, 0.4, 0.5	0.5
M: Dense: units	128, 160, 192	128
M: Dropout: rate	0.3, 0.4, 0.5	0.3

Table 5.7: Hyperparameter selection results using Keras-Tuner RandomSearch algorithm.

Best Configuration		**2nd Best Configuration**		**3rd Best Configuration**	
Score	99,97%	Score	99,95%	Score	99,49%
E: Dense: units	384	E: Dense: units	256	E: Dense: units	192
E: Dropout: rate	0.5	E: Dropout: rate	0.3	E: Dropout: rate	0.2
E: Dense: units	48	E: Dense: units	32	E: Dense: units	64
I: Conv1D: filters	90	I: Conv1D: filters	120	I: Conv1D: filters	120
I: Conv1D: kernels	4	I: Conv1D: kernels	4	I: Conv1D: kernels	5
I: Conv1D: strides	4	I: Conv1D: strides	4	I: Conv1D: strides	4
I: MaxPool1D: pool_size	3	I: MaxPool1D: pool_size	4	I: MaxPool1D: pool_size	4
I: Dropout: rate	0.5	I: Dropout: rate	0.5	I: Dropout: rate	0.4
M: Dense: units	128	M: Dense: units	192	M: Dense: units	128
M: Dropout: rate	0.3	M: Dropout: rate	0.5	M: Dropout: rate	0.4

5.9.2.2 Hybrid Convolutional Neural Network: Experimental Results

In section 5.9.2.1, a brief overview of the explicit and implicit models to be combined are discussed in detail. The decision to employ a hybrid CNN design is based on the work of Yang et al. (2018) who follow a similar structure in analysing explicit and implicit features for an article’s text and image contents.

The data used was split into two subsets; 80% for training the model, and 20% to validate the model during training. The training was performed over 20 epochs. A detailed view of the hybrid CNN is presented in Figure 5.10. The model had 65 548 274 parameters, of which 579 674 were trainable. 64 968 600 parameters were non-trainable. Each epoch took roughly 1 minute and 25 seconds to complete. The Adam optimizer is selected as the optimizer. The hybrid CNN model achieved a weighted accuracy score of 99,66%, 0,0090 for loss, 99,88% as the validation accuracy and 0,0129 as the validation loss. Detailed experimental data can be seen in Appendix B.

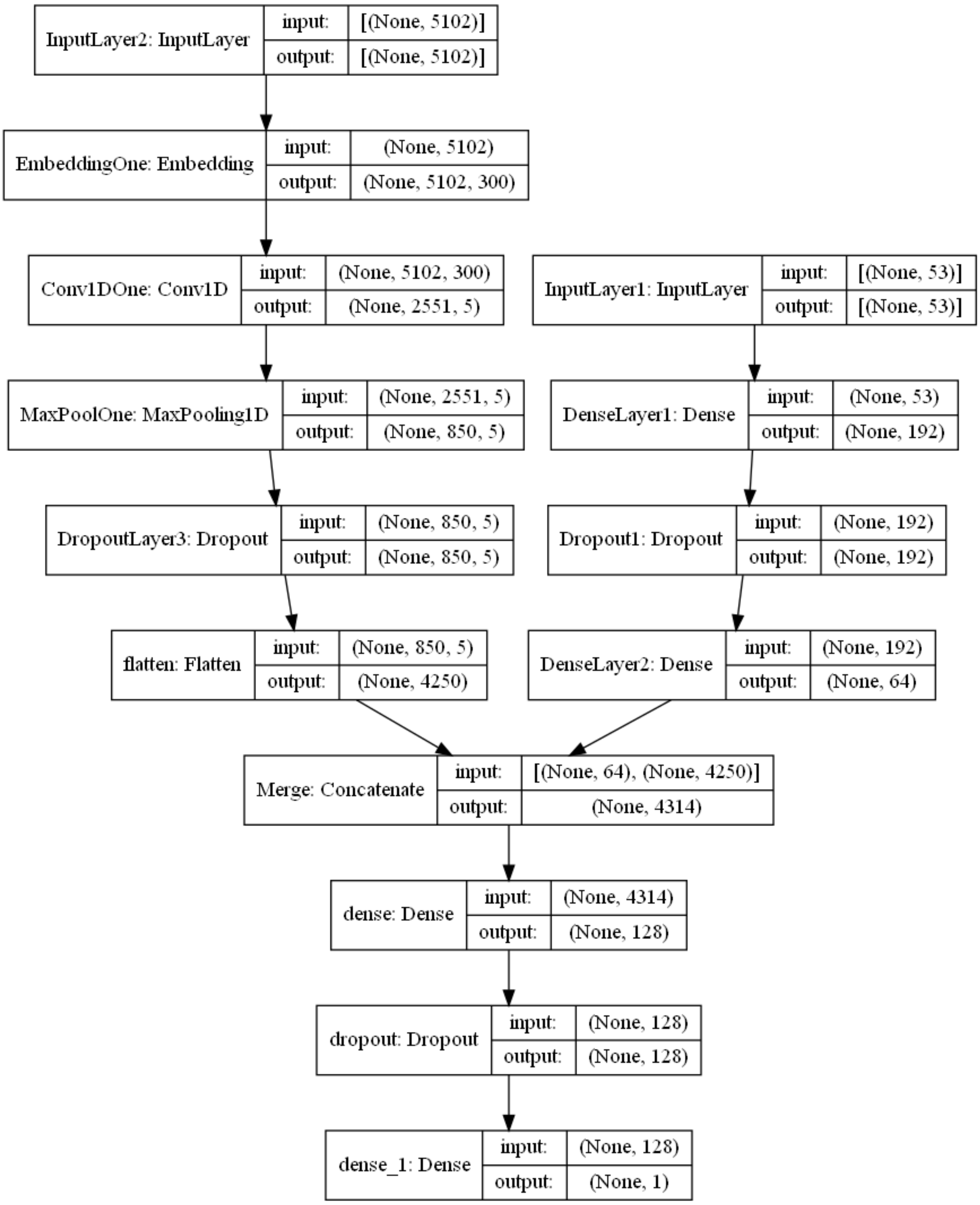


Figure 5.10: Overall architecture of the neural network. The model accepts two inputs, one for the explicit features and one for implicit features extracted by the Convolutional Neural Network.

Due to the problem being modelled as a binary classification problem, the sigmoid function is used on the output layer as the activation function. The SoftMax function is useful in scenarios where the problem is modelled as a multi-class problem. Binary Cross-Entropy is employed as the loss function for the model. Given a range of values, the sigmoid function transforms such values to a figure between 0 and 1 (Sharma,

Sharma, & Athaiya, 2020). If x is the input in numeric form, the sigmoid function is expressed, using equation 22 (Keras, 2020):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{22}$$

The Binary Cross-Entropy function is recommended for binary classification problems. The function calculates loss between the predicted class and the expected class. If y is the class (true or fake news) and $p(y)$ is the predicted class, and N is the total number of samples, the loss function L can be expressed using equation 23 (Agarwal, Mittal, & Goyal, 2020):

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{23}$$

5.9.3 Hybrid Recurrent Neural Network (LSTM) + CNN Model

In the experiment, a hybrid deep learning model for fake news detection is constructed, using Recurrent Neural Network, Convolutional Neural Network and Dense Neural Network architectures. The decision to include the CNN architecture is due to the architecture's ability to extract latent features from data. As part of the RNN architecture, the Long Short-Term Memory cell is used. The decision to follow such a design, and to include RNN technology, is based on an observation made by Nasir et al. (2021) regarding Long Short-Term Memory cells; in a hybrid RNN-CNN neural network model, the LSTM cell could be used in capturing information that relates to the flow of information.

Like the experiment carried out in section 5.8, this experiment uses explicit, handcrafted features, and implicit (sequential) data from the text. The Keras functional API is employed to create a multi-input model; the first model is a Dense Neural Network model, whilst the second model stacks both Convolutional Neural Network and Long Short-Term Memory architecture. The inputs are then concatenated, and a decision is made on the output layer.

5.9.3.1 Hybrid Model: RNN and CNN Details

The Convolutional Neural Network layer is responsible for extracting latent features from the text, whilst the Long Short-Term Memory (LSTM) is responsible for learning from time-series data within the text. All documents are one-hot encoded, using the

Keras Tokenizer API. In terms of the article body, the longest document had 98 words. To ensure all documents have equal lengths, following the one-hot encoding process, zero-padding is employed; zeros are appended to the end (right) of all documents. Figure 5.11 provides a detailed view of the hybrid RNN (LSTM) model.

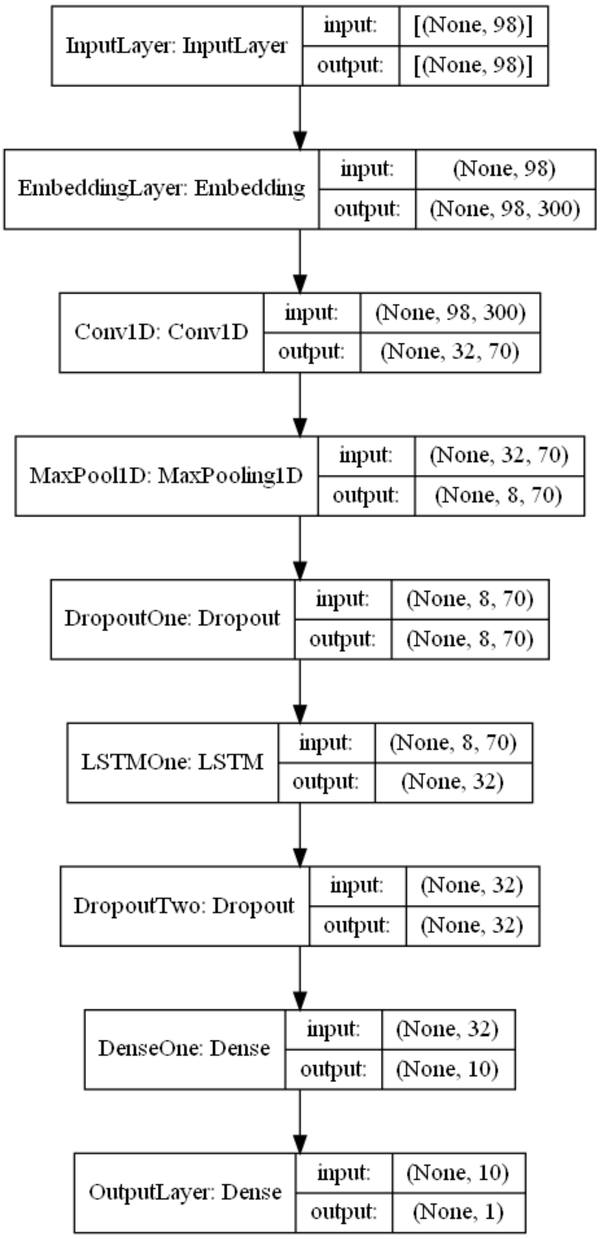


Figure 5.11: Hybrid CNN-RNN model for fake news detection

5.9.3.2 Long Short-Term Memory Cell Configuration

For the LSTM cell, the configuration is kept in line with the documentation specified by Keras. When using an Nvidia GPU to build the RNN, Keras can utilise Nvidia CuDNN technology, which yields faster training times (Zhu & Chollet, 2019). As per the Keras documentation, the activation function is set to tanh, recurrent dropout is kept at 0, use bias is set to true and unroll is set to False.

A pre-trained word2vec model is employed to generate word embeddings for all words contained in the corpus. The Google News pre-trained word2vec model is selected. For each word sent to the word2vec model, a 300-dimensional vector space is generated. Words that are not found in the word2vec model's dictionary are replaced with 0. A total of 19 876 words were not found; such words were misspelt words or slang. On the Keras Random Search initializer, the max_trials parameter is set to 3 and the executions_per_trial parameter is set to 2

5.9.3.3 Hybrid Model: RNN and CNN Hyperparameter Selection

Like in section 5.9.1, the Keras-Tuner library is selected for determining the best model configuration. The experiment uses the Random Search technique to determine the best model configuration. The purpose of the model is to make a prediction on the data, whilst considering latent text features and features describing long-term dependencies. Table 5.8 illustrates parameters explored in the Random Search. The best configuration is used for the final model, as shown in Table 5.9.

Table 5.8: Hyperparameter selection criteria used by Keras Random Search technique

Layer	Configuration	Best Configuration
Conv1D	Filters: 50, 70, 90 Kernel Size: 3, 4, 5 Strides: 3, 4, 5	Filters: 50 Kernel Size: 3 Strides: 4
MaxPool1D	Pool Size: 4, 5, 6	Pool Size: 3
Dropout	Rate: 0.2, 0.3, 0.4	Dropout: 0.3
LSTM	Units: 16, 24, 32	Units: 32
Dropout	Rate: 0.2, 0.3, 0.4	Dropout: 0.2
Dense	Units: 6, 10, 14	Units: 14
Adam Optimizer: Learning Rate	Rate: 0.001, 0.0005, 0.005	Learning Rate: 0.001

Through the Random search process, the best configuration achieves a score of 89.49%. The 2nd and 3rd best configurations were close to the best configurations, with scores of 89.38% and 89.49% respectively. Table 5.9 shows a detailed breakdown of scores attained and selected hyperparameter configuration, following the Random Search.

Table 5.9: Hyperparameter selection processing, using Random Search functionality in the Keras-Tuner library.

3rd Best Configuration		**2nd Best Configuration**		**Best Configuration**	
Score	86.83%	Score	89.38%	Score	89.49%
Conv1D: filters	50	Conv1D: filters	50	Conv1D: filters	70
Conv1D: strides	4	Conv1D: strides	3	Conv1D: strides	3
Conv1D: kernel_size	3	Conv1D: kernel_size	3	Conv1D: kernel_size	5
MaxPooling1D: pool_size	5	MaxPooling1D: pool_size	4	MaxPooling1D: pool_size	4
Dropout: rate	0.3	Dropout: rate	0.2	Dropout: rate	0.3
LSTM: units	24	LSTM: units	32	LSTM: units	32
Dropout: rate	0.2	Dropout: rate	0.2	Dropout: rate	0.3
Dense: units	14	Dense: units	14	Dense: units	10
Adam: learning_rate	0.001	Adam: learning_rate	0.001	Adam: learning_rate	0.001

5.9.3.4 Hybrid Recurrent Neural Network (LSTM) Results

The data used in the hybrid RNN model was split into an 80% training and 20% training set. The hybrid model was trained over 20 epochs, with the batch size set to 64. The model had a total of 15 522 395 parameters, 118 595 of which were trainable, whilst 15 403 800 were non-trainable parameters. The Adam optimizer is selected, with a learning rate of 0.001. The hybrid RNN model attains a weighted accuracy score of 88,81%, loss of 25,43%, validation accuracy of 88,86% and a validation loss of 25,81%. A detailed view of results obtained at each epoch can be seen in Appendix E.

5.9.4 Convolutional Neural Network Experiment

In this experiment, the task of fake news detection is explored using Convolutional Neural Network architecture. Convolutional Neural Networks are useful in extracting meaningful features from data. Like other neural network experiments, the hyperparameter selection process is undertaken using the Keras-Tuner library. The neural network is constructed using the Keras Functional API. Figure 5.12 illustrates a high-level view of the proposed model.

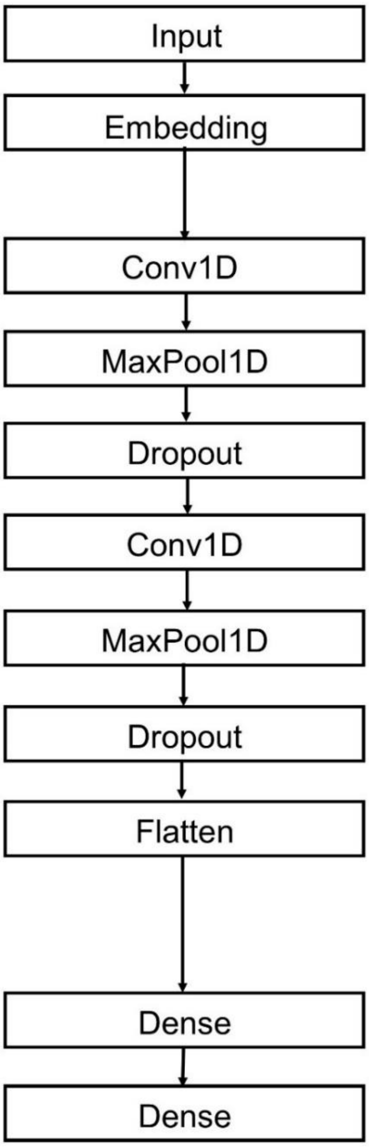


Figure 5.12: Overview of proposed CNN model for fake news detection

5.9.4.1 Convolutional Neural Network Experiment: Hyperparameter selection

Before training and testing any given model, it is vital to determine the optimal hyperparameters systematically. As such, this experiment uses the Keras-Tuner library to determine the best model configuration. The Random Search technique is selected, where combinations of parameter configurations are selected and tested over a certain number of trials. The number of trials is set to 3, and the number of executions per trial is set to 2. Table 5.11 shows parameter configurations selected together with the best parameter values, as determined by the Random Search technique in the Keras-Tuner library. The Input and Embedding layers are not included in the parameter selection process. The hyperparameter selection process is performed over 20 epochs. The final output layer has 1 unit, and the sigmoid activation function is employed. The model is compiled using the Adam optimizer, with the Binary Cross-Entropy function as the loss function. Basic data pre-processing techniques outlined in section 5.5 are employed. Data is split into 80% training set, and 20% testing set. Table 5.11 shows the top 3 configurations explored using Keras random search technique. The best configuration provides optimal results, scoring 99,44%.

Table 5.10: Parameter values explored in the hyperparameter selection process

No.	**Layer and Parameter**	**Values Tested**	**Best Value**
1	Conv1D: filters	70, 90, 120	70
1	Conv1D: kernel_size	2, 3, 4	3
1	Conv1D: strides	2, 3, 4	4
2	MaxPool1D: pool_size	2, 3, 4	2
3	Dropout: rate	0.3, 0.4, 0.5	0.5
4	Conv1D: filters	30, 50, 70	70
4	Conv1D: kernel_size	2, 3, 4	4
4	Conv1D: strides	2, 3, 4	4
5	MaxPool1D: pool_size	2, 3, 4	3
6	Dropout: rate	0.3, 0.4, 0.5	0.3
7	Dense: units	16, 32, 64	64
	Adam Optimizer: learning_rate	0.001, 0.0001, 0.0005	0.001

Table 5.11: Results obtained at each trial, after the hyperparameter selection process

Top Result		**2nd Best Result**		**3rd Best Result**	
Score	99,48%	Score	99,43%	Score	99,22%
Conv1D: filters	70	Conv1D: filters	110	Conv1D: filters	110
Conv1D: kernel_size	3	Conv1D: kernel_size	3	Conv1D: kernel_size	2
Conv1D: strides	4	Conv1D: strides	4	Conv1D: strides	4
MaxPool1D: pool_size	2	MaxPool1D: pool_size	4	MaxPool1D: pool_size	2
Dropout: rate	0.5	Dropout: rate	0.5	Dropout: rate	0.3

Table 5.12: Results obtained at each trial, after the hyperparameter selection process (continued)

Conv1D: filters	70	Conv1D: filters	30	Conv1D: filters	30
Conv1D: kernel_size	4	Conv1D: kernel_size	4	Conv1D: kernel_size	3
Conv1D: strides	4	Conv1D: strides	3	Conv1D: strides	3
MaxPool1D: pool_size	3	MaxPool1D: pool_size	3	MaxPool1D: pool_size	2
Dropout: rate	0.3	Dropout: rate	0.5	Dropout: rate	0.4
Dense: units	64	Dense: units	32	Dense: units	16
Adam Optimizer: learning_rate	0.001	Adam Optimizer: learning_rate	0.001	Adam Optimizer: learning_rate	0.001

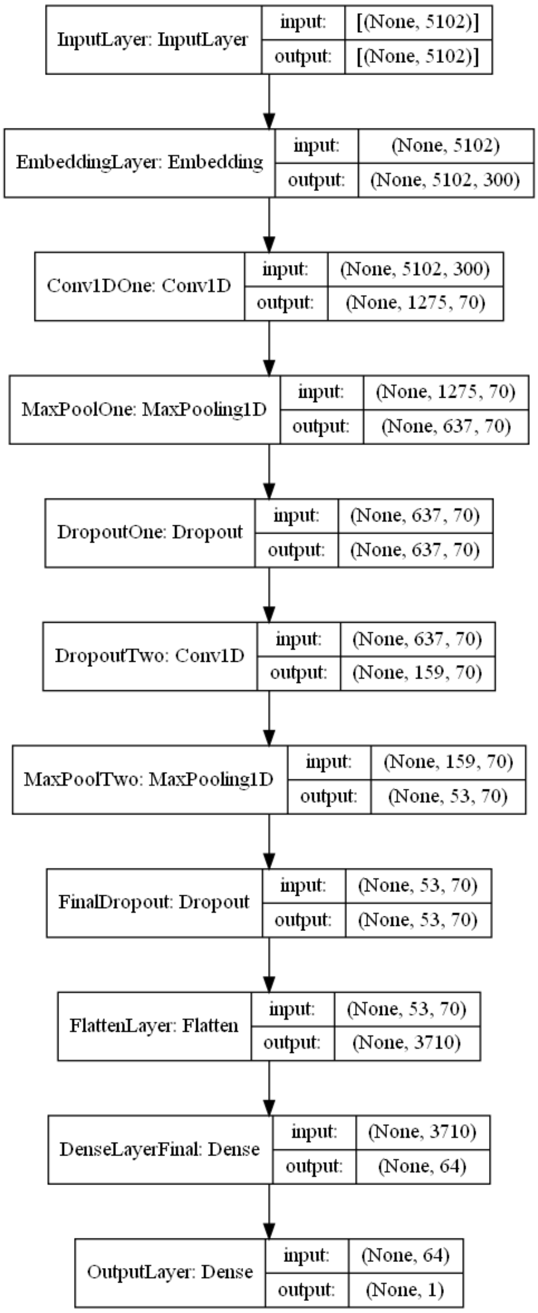
5.9.4.2 Hyperparameter selection training and validation scores

The tables contained in this section outline results obtained at each trial of the hyperparameter selection process. The loss, accuracy, validation loss and validation accuracy metrics are selected. The training was performed over 20 epochs. Owing to the CNN architecture not being computationally expensive, no batch size was specified. The batch size parameter is useful in scenarios where memory resources are limited. Furthermore, details about the number of parameters, trainable and non-trainable, are reported in this section.

5.9.4.3 Convolutional Neural Network – Experimentation Results

Using the best configuration in Table 5.11, the model is trained and evaluated over 20 epochs. Figure 5.13 illustrates the final design of the convolutional neural network model. The final model has 65 288 609 parameters. 320 309 were trainable parameters and 64 968 300 are non-trainable parameters. The CNN model achieves a weighted accuracy score of 98,61%, loss of 3,50%, validation accuracy of 98,85% and a validation loss score of 3,72%. Detailed training and testing results obtained at each epoch can be found in Appendix C.

Figure 5.13: Convolutional Neural Network (CNN) design, using the best configuration determined by the hyperparameter selection process



Like the previous deep learning experiments, a pre-trained word2vec model is used to generate word embeddings, which are used as weights for the model. The Keras Tokenizer API is used to encode documents contained in the dataset through the process of one-hot encoding – each unique word is assigned a numeric value. Each occurrence of a word contained in the vocabulary is replaced with its corresponding numeric value. The encoded documents are used as inputs into the model. An OOV (Out of Vocabulary) value is assigned for words that are not contained in the word vocabulary. The longest document found in the dataset contained 5102 words. Data pre-processing techniques outlined in this chapter are employed in this experiment.

5.9.5 Recurrent Neural Network Experiment – Gated Recurrent Unit

In this experiment, a recurrent neural network using the Gated Recurrent Unit (GRU) cell is constructed. Like the hybrid model constructed in section 5.9.2, this model uses summarized texts generated using Hugging Face's T5 transformers library. The decision to use document summaries is to address the limited memory and computing power present on the computer used for the experiments. Summarized texts are encoded using Keras Tokenizer API, and word embeddings are generated using a pre-trained word2vec model. To use the optimized, cuDNN implementation of the GRU cell, basic layer configuration is kept in line with Keras documentation; activation function set to tanh, recurrent activation set to sigmoid, recurrent dropout set to 0, use bias set to True, reset after set to True, and unroll set to False (Keras, 2021). The model is constructed using the Keras Functional API (Chollet, 2019). A basic overview of the model is presented in figure 5.14.

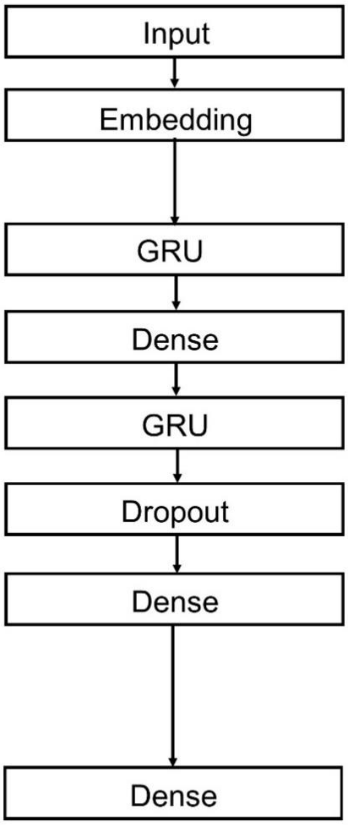


Figure 5.14: Recurrent Neural Network, using GRU cells.

5.9.5.1 RNN Experiment – Hyperparameter Selection

Before training and testing any given model, the best model configuration should be determined systematically. As such, this experiment uses the Keras-Tuner library to determine the best model configuration. The Random Search technique is selected, where combinations of parameter configuration are selected and tested over a certain number of trials. The number of trials is set to 3, and the number of executions per trial is set to 2. Table 5.13 shows parameter configurations selected together with the best parameter values, as determined by the Random Search technique in the Keras-Tuner library. The hyperparameter selection process is performed over 20 epochs. The final output layer has one unit, and the sigmoid activation function is employed. The model is compiled using the Adam optimizer, with the Binary Cross-Entropy function as the loss function. Data is split into an 80% training set, and a 20% testing set. Table 5.13 and Table 5.14 illustrate the parameters selected and the results obtained given the configuration selected by the Keras Random Search technique. The best configuration scores at 89,33%.

Table 5.13: Summary of parameters explored during the hyperparameter selection process.

No.	Layer and Parameter	Tested Values	Best Values
1	GRU: units	64, 96, 128	96
2	Dropout: rate	0.3, 0.4, 0.5	0.3
3	GRU: units	24, 32, 48	48
4	Dense: units	8, 16, 24	16
5	Dropout: rate	0.3, 0.4, 0.5	0.3
	Adam Optimizer: learning rate	0.01, 0.05, 0.005	0.005

Table 5.14: Summary of results and configuration used at each trial of the Random Search

Top Result		2nd Best Result		3rd Best Result	
Score	89,33%	Score	88,25%	Score	86,86%
GRU: units	96	GRU: units	64	GRU: units	96
Dropout: rate	0.3	Dropout: rate	0.3	Dropout: rate	0.4
GRU: units	48	GRU: units	24	GRU: units	32
Dense: units	16	Dense: units	24	Dense: units	24
Dropout: rate	0.3	Dropout: rate	0.3	Dropout: rate	0.5
learning rate	0.005	learning rate	0.01	learning rate	0.01

5.9.5.2 RNN (GRU) Experiment – Results

Using the summarized documents dataset generated from original articles, and the hyperparameter configuration deemed best by the Keras-Tuner Random Search technique, the tables that follow represent the results obtained whilst training and validating the model. The model was trained over 20 epochs, with the dataset being split into an 80% training set, and 20% validation set. Figure 5.15 illustrates the final model design. Generated word embeddings are used as weights for the model, and encoded documents are used as inputs for the model. The longest summary contained 98 words. Like other neural network models in the series of experiments contained in this chapter, the sigmoid function is applied at the last output layer. The RNN (GRU) model attains a weighted accuracy score of 89,70%, a loss of 24,10%, validation accuracy of 88,79% and validation loss of 26,04%. Detailed training and testing data for the RNN (GRU) model can be found in Appendix D.

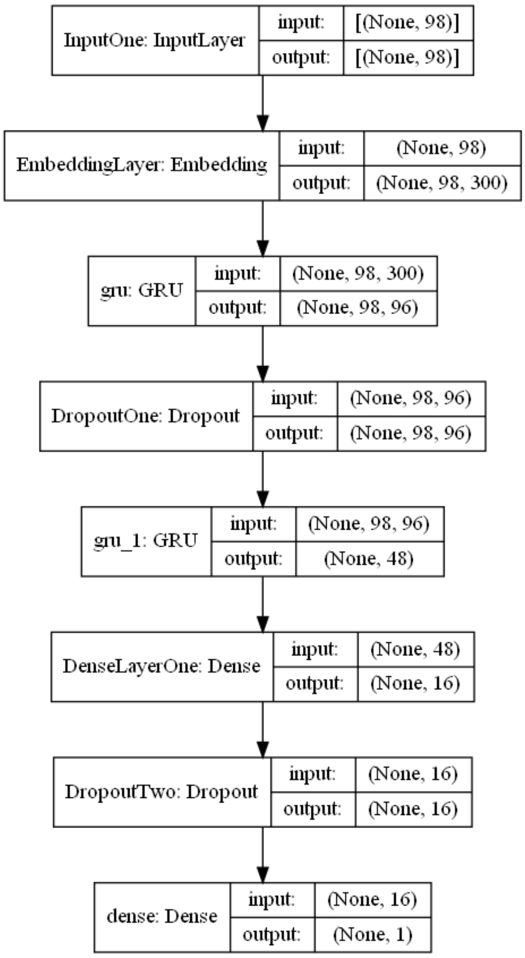


Figure 5.15: Final GRU model, using the configuration deemed best by Keras-Tuner Random Search

5.10 Summary of Experimental Results

In the preceding sections of this chapter, numerous deep learning and machine learning models were selected for the detection of text-based fake news. Before the execution of the selected models, a feature set that best embodies the indicators of fake news was selected. A summary of these indicators, and how they map to the feature set, is presented in Table 5.15. A summary of the selected deep learning models and the weighted performance results are shown in Tables 5.16 and 5.17.

5.10.1 Indicators of Fake News

The table below shows the selected indicators of text-based fake news. For each indicator, one or more features that map to the indicator are listed.

Table 5.15: Summary of indicators of text-based fake news and the corresponding features

Text-Based Fake News Indicator	**Mapped Features**
Fake news articles generally contain short article bodies and longer article titles.	Title word count
	Article body word count
	Average sentence length
Fake news articles generally use simpler sentence structures.	Title punctuation count
	Article body punctuation count
	Parts of speech tag counts
	Gunning Fog Index
	Flesch Kincaid Grade Level Index
	Automated Readability Index
	Type-Token Ratio
Fake news article’s generally have headlines which aren’t related to the article body.	Cosine Similarity

To transform the text into logical, numeric representations where relationships between words are kept, a Doc2vec model is trained and used to generate document features for each article.

5.10.2 Summary of Classification Model Results

This section provides a summary of the results obtained, using the feature set outlined in Table 5.2. For each deep learning model, an average accuracy, precision, and recall score are provided. Training and testing are performed over 20 epochs, using optimal model configurations determined by hyper-parameter tuning processes. Detailed performance results are included in Appendices B, C, D and E. Using common performance metrics such as precision, recall, Area Under the Curve (AUC) and F1-Score, Table 5.17 provides detailed performance results for the selected deep learning models in the experiments. The performance results reported in Table 5.17 consider the test portion of the selected dataset (20% of data). The results presented in this section are revisited in section 5.11, where a detailed analysis of the results is examined.

Table 5.16: Summarized training and validation results for the selected deep learning models

Deep Learning Model	Accuracy	Loss	Validation Accuracy	Validation Loss
Hybrid CNN Model	99,66%	0,0090	99,88%	0,0129
Hybrid RNN (LSTM) - CNN Model	88,81%	0,2543	88,86%	0,2581
CNN Model	98,61%	0,0350	98,85%	0,0372
RNN (GRU) Model	89,70%	0,2410	88,79%	0,2604

Table 5.17: Weighted Precision, Recall, F1-Score and AUC metrics for the selected neural network models.

Deep Learning Model	Precision	Recall	F1-Score	AUC
Hybrid CNN Model	99,95%	99,81%	99,88%	100,00%
Hybrid RNN (LSTM) - CNN Model	87,53%	91,19%	89,32%	96,04%
CNN Model	98,25%	99,59%	98,91%	99,83%
RNN (GRU) Model	87,73%	91,48%	89,57%	95,92%

Table 5.18: Summary of performance results of each machine learning classifier

Classifier	Precision	Accuracy	Recall	F- Measure	ROC
SVM	99,30%	99,40%	99,40%	99,40%	99,30%
DT	95,00%	94,90%	95,30%	95,20%	94,80%
RF	99,40%	98,90%	98,40%	98,90%	99,90%
KNN	96,40%	91,20%	86,40%	91,20%	91,50%
XGB	99,10%	98,80%	98,60%	98,80%	98,80%
AB	99,30%	99,30%	99,40%	99,30%	99,30%

5.11 Discussion

In the preceding sections, fake news detection using machine learning and deep learning techniques were explored, using several strategies and models. In this section, a brief discussion of the results exhibited in the preceding sections is presented.

In the experiments, various machine learning classifiers and deep learning models are selected for the task of automated fake news detection. Such selections are primarily based on the works of other researchers who have explored automated fake news detection using a variety of machine learning and/or deep learning models. In the machine learning portion of experiments, this study selects a combination of readability metrics, vector representation of documents, and accumulative features that describe sentence structure and composition. These features are fed into various machine learning classifiers after optimal configurations for each classifier are established.

In a deep learning context, a selection of hybrid deep learning models and base deep learning models are selected. In the experiment, a hybrid deep learning model is constructed, using convolutional neural network technology, and fully connected dense layers. The hybrid CNN model attains fewer false positives and false negatives compared to the selected machine learning classifiers. Additionally, the hybrid model attains higher true positives and true negatives than the machine learning classifiers. These observations are an indicator that deep learning models, specifically hybrid models, can achieve superior results compared to machine learning classifiers. Such an observation is supported by experimentation results reported by Cuevas et al. (2020). The authors select neural network models such as Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and several machine learning classifiers, namely, Decision Tree, Random Forest, K-Nearest Neighbour, Support Vector Machine, Xtreme Gradient Boost. Using Precision, Recall, F1-Score, False Positive Rate and False Negative Rates as metrics, the authors achieve better results through the use of deep learning architecture compared to traditional machine learning classifiers (Cuevas et al., 2020).

Recurrent Neural Networks (RNN) are known to use more computational resources and are slower to train. This statement is supported by an observation made by Mittal & Umesh (2020) who state that recurrent neural networks use more memory due to the model requiring more training epochs to converge. Due to limited computational resources on the notebook used to carry out experiments, text summarization is employed, to reduce the number of parameters whilst considering articles in their entirety. The Hugging Face implementation of the T5 framework is selected to summarize documents contained in the fake and real news datasets. The summarized documents are used in an RNN that uses Gated Recurrent Unit (GRU) cells, and a hybrid neural network that stacks LSTM and CNN layers. Though the results obtained fall short when compared to the hybrid models and base CNN experiments, they do show that summarized texts and deep learning models can achieve promising results in classification problems. The experiments show that text summarization could be one of many options for researchers and developers with limited computational resources, though the use of such an approach may depend on the expected outcomes and use-cases of the model. In cases where computational resources are limited, it may be worthwhile exploring several strategies, for example, using cloud providers such as Google or Microsoft, which often have more resources specific to machine learning and deep learning workflows.

In terms of the stacked CNN-LSTM model compared to the GRU model, the GRU model converged quicker than the CNN-LSTM model. Furthermore, from epoch 10 of the results obtained through the GRU model, a general decrease in accuracies and an increase in loss is observed. This observation is a sign the model could be overfitting, and strategies such as early stopping or including more training epochs could overcome this. Additionally, experimentation could be extended to include exploring the effects varying optimizers such as Stochastic Gradient Descent, RMSProp, and AdaGrad have on the models contained in the experiment. Through experimentation, Bahad, Saxena, & Kamal, (2020) select RMSProp, AdaGrad, and the Adam optimizers, and find the RMSProp optimizer provides the best model performance (Bahad et al., 2020). The hybrid, stacked CNN-LSTM model continuously improved over 20 epochs, with a general decline in loss and a general increase in accuracy.

The hybrid CNN model depicted in section 5.9.2 considers manually created textual features, as well as implicit features present in the text. Given the hybrid CNN model depicted in section 5.9.2 attains the best overall results, Though the hybrid model in section 5.9.2 considers just textual data and not images and other metadata associated with articles, it can be deduced that hybrid approaches can deliver promising results.

In all experiments carried out, vectors that best represent words, relationships between words given a certain context, and vectors that best represent documents, are selected. The performance-related results attained by each of the selected machine learning algorithms and deep learning models prove the efficacy of generating features through pre-trained models within the transfer learning domain (word2vec, doc2vec, GloVe, etc.). Though pre-trained models such as word2vec, doc2vec, and GloVe exist and can retain relational information between words, such models do not necessarily outperform simpler word representation methods such as Bag of Words or TF/TF-IDF features. Through a series of experiments on various machine learning classifiers, Silva et al. (2020) found models which used word2vec and FastText word embeddings produced results that were as much as 11% lower than models which used Bag of Words features. The authors further add that such an observation could be the result of word embeddings models being trained on clean text, whilst fake news articles often contain noise such as slang and misspelt words. In light of the experiments contained in this chapter, the explanation presented by Silva et al. (2020) could be supported by the high number of words not found in the word2vec model – a total of 19 876 words for the summarized articles derived from the dataset authored by Bisaillon (2020), and a total of 161 846 words for the full Fake Real News dataset.

Table 5.17 presents a detailed view of the performance results for the selected deep learning architectures selected for the experiment. The precision, recall, AUC and F1-score metrics are selected. The results reported in Table 5.17 consider the test portion of the fake and real news articles dataset curated by (Bisaillon, 2020). The numbers reported in this table are a weighted average between all 20 epochs. The best performing model is the Hybrid CNN model, with a precision of 99,95%, recall of 99,81%, F1-Score of 99,88% and an AUC score of 100%. The worst performing model was the Hybrid RNN (LSTM) – CNN Model, with a precision score of 87,53%, recall of 91,19% and an F1-score of 89,32%. The fact the Hybrid CNN model, which uses explicit and implicit text features yielded the best performance results can be supported by the model achieving the lowest weighted validation loss of 0,0129. This observation is further supported by the model attaining fewer false positives and fewer false negatives, and higher true positives and true negatives. These observations further prove hybrid neural network models which combine explicit text features and implicit text features can produce promising results. The results exhibited by the worst performing model could be attributed to the model attaining the worst validation loss of 0,2581, and that some vital textual information may have been lost as a result of text summaries being used.

Though the experiments contained in this chapter demonstrate that promising results are achievable through analysing the textual aspect of fake and real news articles, the models could be extended to include other information about the articles, such as media (images, video), information about the source and information about the author. In reality, most articles often contain images and video. Yang et al. (2018) construct a hybrid TI-CNN (Text and Image Convolutional Neural Network) which considers textual and image data contained in fake and real news articles. The TI-CNN model outperforms other vanilla models which consider text or image data (Yang et al., 2018).

Finally, the accepted definition of fake news can have a significant influence on the experiment design and results. Whilst several works consider fake news as a binary classification problem (an article can be truthful or false), the definition of fake news could be refined into subcategories, such as propaganda, satire, etc. For training purposes, the Doc2Vec model, a subset of the FakeNewsCorpus dataset authored by Szpakowski (2018) is selected. The first 12 000 articles, in the 12 categories of articles present in the FakeNewsCorpus dataset is selected. In addition, the ClaimBuster project analyses statements through traditional machine learning approaches and classify the statements into three categories (Hassan et al., 2017). The Fake Real News dataset by Bisaillon (2020) has been used by several authors for the task of fake news detection. Table 5.19 shows publications that used the ISOT dataset, and the results obtained. The results obtained from the selected machine learning and deep learning models in Table 5.16, Table 5.17 and Table 5.18 are within margin of results obtained by other authors in Table 5.18. The hybrid CNN model constructed in section 5.9.2 outperforms the accuracy result obtained by Ahmed, Traore, & Saad (2017) as listed in Table 5.18, with a difference of 0,66%. From a precision and and recall perspective, the hybrid CNN model outperforms the precision and recall results obtained by Nasir, Khan, & Varlamis (2021) as listed in Table 5.18, with differences of 0,95% and 0,81% respectively. For simplicity, the best results from each study are selected. For studies where certain performance metrics are not provided, these are marked with a dash symbol

Table 5.19: Comparison of results obtained by other researchers using the Fake Real News Dataset by Bisaillon (2020)

Author	Accuracy	Precision	Recall	F1	Model
Ahmed, Traore, & Saad (2017)	92%	-	-	-	Linear SVM
Jiang, Ping Li, Ul Haq, Saboor, & Ali (2021)	99.94	100%	100%	100%	Stacking Model (CNN, LSTM, GRU + DT, RF, KNN, LR, SVM + RF)
Kula, Choras, Kozik, Ksieniewicz, & Wozniak (2020)	99,86%	99,82%	99,91%	>90%	RNN (LSTM)

Table 5.20: Comparison of results obtained by other researchers using the Fake Real News Dataset by Bisaillon (2020) (continued)

Nasir, Khan, & Varlamis (2021)	99%	99%	99%	99%	Hybrid CNN-RNN
Goldani, Safabakhsh, & Momtazi (2021)	99,9%	-	-	-	CNN (with margin loss)

5.12 Conclusion

The purpose of experimentation in the research project is to test various methods for fake news detection, using machine learning and deep learning approaches reported by other researchers in the field of fake news. Whilst the technology to carry out automated fake news detection is available, it is also important to understand what makes fake news fake news, and to determine indicators that could suggest an article is a fake news article. To that end, the experiments use features that are based on observations made by other researchers as to what best separate fake news from real news articles. The results obtained by all models show the effectiveness of such indicators through both machine and deep learning approaches. The selected models could be improved and possibly extended to capture more key information that could be used for the task of detecting fake news.

Using the foundation outlined in the literature related to machine learning in Chapter 3, the indicators defined in this chapter, and the common differences found in fake and real news articles, this chapter addresses secondary objective 3:

3. *Validate the identified indicators through the application of selected machine learning algorithms.*

Chapter 6 presents guidelines for automated fake news detection. The experimental results obtained in this chapter, and observations from literature found in chapters 2 and 3, serve as the foundational basis for constructing the guidelines.

6 Chapter 6 - Guidelines For the Detection of Online Text-Based Fake News

This chapter presents a collection of guidelines that could be used in the development of machine learning and deep learning models for the task of automated fake news detection. The foundational basis supporting such guidelines is derived from observations noted in the experiments undertaken in Chapter 5, and other indicators highlighted by researchers in the fake news detection realm. The guidelines aim to address three key areas, namely, the data aspect, the fake news detection model aspect, and the evaluation aspect. Each guideline describes the problem, presents a solution, and mentions other points of consideration.

6.1 Implementation Details

The guidelines described below were implemented using the Python programming language. A series of Python libraries were used for various aspects of the process, such as data cleaning, classifier performance optimization, and performance evaluation. For this study, SciKit-Learn, Keras, Keras-Tuner, Pandas, NLTK, Py-readability-metrics and lexical-richness libraries were used. The decision to use Python and the mentioned Python libraries are based on the popularity, widespread support, and maturity of Python for research. The decision to use Python libraries such as py-readability-metrics and lexical-richness to calculate text readability and structure is due to the ease of implementation in an existing machine learning workflow, and the fact that mathematical calculations are already implemented within these libraries. The widespread use of these tools by other researchers in the fake news detection community further solidifies the use of these libraries. Ahmed, et al. (2019) recognise the popularity of Python due to its simplicity and wide availability of libraries for data science.

Though other programming languages provide their machine learning frameworks, such as Microsoft's ML.NET framework (Ahmed, et al., 2019), these frameworks are relatively new, so have not reached the level of maturity seen with Python and its various packages designed for data science and machine learning. Microsoft launched the ML.NET framework in 2019, to provide .NET developers with a machine learning implementation that can be integrated with existing .NET applications (Ahmed, et al., 2019).

A summary of the Python packages used for the experiments are described below:

1. **SciKit-Learn:** A popular Python-based machine learning framework, which was authored in 2007.
2. **Keras:** A simple-to-use Deep Learning library, which is built on top of the TensorFlow 2.0 libraries.
3. **Keras-Tuner:** A performance optimization library used for deep learning models built with Keras.
4. **Pandas:** A popular data analysis and data manipulation Python library (Pandas, 2012)
5. **NLTK (Natural Language Tool Kit):** A popular Python package for working with textual language data (NLTK, 2021).
6. **Py-readability-metrics:** A Python module that calculates the readability of texts, using 9 readability metrics, namely, Flesch Kincaid Grade Level, Flesch Reading Ease, Dale Chall Readability, Automated Readability Index, Coleman Liau Index, Gunning Fog, SMOG, Spache and Linsear Write (DiMascio, 2019).
7. **Lexical-richness:** A Python library that calculates the richness of texts using various metrics, such as the unique term count, Type-Token Ratio (TTR), Root Type-Token Ratio (RTTR), Corrected Type-Token Ratio (CTTR), mean-segmental type-token ratio (MSTTR), Moving-Average Type-Token Ratio (MATTR), Measure of Textual Lexical Diversity (MTLD), and Hypergeometric Distribution Diversity (HD-D) (Shen YS, 2018).

6.2 Guidelines for Fake News Detection

This section presents the guidelines resulting from this study. For each guideline, a brief description of the problem, a solution, and possible constraints are provided. The problem definition describes key issues which should be addressed when developing systems for text-based fake news detection. The solution describes a certain method of resolving the identified text-based fake news problem. Finally, to demonstrate the practicality and effectiveness of the solution, code examples through pseudocode and Python code are presented.

6.2.1 Define an Applicable Definition of 'Fake News'

6.2.1.1 Problem Description

Various organisations and authors have differing definitions of what is considered 'fake news'. A standardised definition of fake news is not slated in the field of misinformation detection. In a practical sense, an online news reader might have a different view of what is considered 'fake news' over other readers. Fernandez (2017) describes fake news as misleading news articles which lack a factual foundation. Ahmed et al. (2017) describe fake news as another category of spam. Wasserman (2017) describes how satirical shows and websites have been labelled as sources of fake news, though not in a negative connotation. The accepted definition of fake news can influence the designs of systems built for online fake news detection.

6.2.1.2 Solution

Before attempting any experimentation, an acceptable definition of fake news, related to the study, must be defined. The selected definition will affect other decisions such as the data selection, the number of categories, the selected models and/or performance metrics. In Chapter 5, articles are fit into two broad categories, truthful or false.

6.2.1.3 Example 1

In a project aimed at verifying statements from presidential debates, Hassan, Arslan, Li, & Tremayne (2017) define spoken statements as being one of three possible categories:

1. Non-Factual Sentence (NFS)
2. Unimportant Factual Sentence (UFS)
3. Check-Worthy Factual Sentence (CFS)

6.2.1.4 Example 2

In the work of Shu, Mahudeswaran, & Liu (2018), the authors define fake news as a binary classification problem. Therefore, articles are split into two categories:

1. Fake
2. Truthful

6.2.2 Select Appropriate Data for the Task of Fake News Detection

6.2.2.1 Problem Definition

Presently, various datasets for fake news detection exist. Such datasets vary in language, lengths of text, size of the dataset, the age of datasets, and additional data fields. An applicable dataset will depend on the accepted definition of 'fake news' and other permutations surrounding the classification problem. Selecting datasets with a limited number of samples, or articles that are not long enough may lead to poor classification results. Jain & Kasbe (2018) suggest that the use of a larger dataset, greater than 11 000 articles as stated in their experiment, could improve learning, because of the larger availability of news content.

6.2.2.2 Solution

Select a dataset that contains a balanced number of articles across all fake news categories and contains a large number of samples. Articles should be closely related to real-world articles in terms of article lengths. The recommendations made by Jain & Kasbe (2018) include the use of datasets that contain lengthy articles. Part of the basic dataset requirements for fake news detection outlined by Conroy et al. (2015) includes datasets containing articles that have similar article lengths.

For multi-class fake news detection problems, the dataset authored by Szpakowski (2018) is recommended. The dataset contains full news articles, spread over 11 categories of fake news. Furthermore, the dataset contains 9 408 908 articles (Szpakowski, 2018). For binary fake news classification scenarios, the dataset authored by Bisaillon (2020) is recommended. The dataset contains 23 481 fake news articles and 21 417 real news articles.

Data visualization techniques, such as creating word clouds using the wordcloud (Mueller, 2020) and matplotlib Python libraries, can assist with illustrating word importance and word distributions among the varying article classes.

To demonstrate the practical implementation of this guideline, Figure 6.1 shows the pseudocode required to set up word cloud generation. Figure 6.2 shows the Python code required to set up word cloud generation.

```
START
define function generate_word_cloud():
    initialize empty fake news dataset
    initialize empty real news dataset
    for each index and row in articles:
       if the label of the current article is real news:
          add article text to real news dataset
       else:
          add article text to fake news dataset
    initialize word vectorizer for fake news articles
    get fake news word vectors by fitting the fake news set into the word
vectorizer
    initialize real news word vectorizer for real news articles
    get real news word vectors by fitting the real news set into the real
news word vectorizer
    initiate an empty fake news word frequency object
    initiate an empty real news word frequency object
    for each word and index in fake news word vectorizer’s vocabulary:
       calculate the word frequency of the current word by adding up all
values in the current column of the fake news vectors
       add word frequency total to fake news word frequencies object

    for each word and index in the real news word vocabulary:
calculate the word frequency of the current word by adding up all values
in the current column of the real news vectors
            add word frequency total to real news word frequencies object

      create an empty word cloud using fake news word frequencies
      plot the fake news word cloud
      save fake news word cloud as an image

      initialize empty word cloud using real news word frequencies
      plot the real news word cloud
      save the real news word cloud as an image
END
```

Figure 6.1: Pseudocode example for the word cloud generation

```
def generate_word_cloud(self):

    fake_news_data = DataFrame()
    real_news_data = DataFrame()

    for index, row in self.data_frame.iterrows():
        if not isinstance(row["text"], str):
            continue
        if len(row["text"]) == 0:
            continue
        if row["label"] == 0:
            newRow = {"title": row["title"],
                      "text": row["text"],
                      "label": row["label"]}
            real_news_data = real_news_data.append(newRow,
ignore_index=True)
        else:
            newRow = {"title": row["title"],
                      "text": row["text"],
                      "label": row["label"]}
            fake_news_data = fake_news_data.append(newRow,
ignore_index=True)

    word_vectors = TfidfVectorizer(lowercase=True, analyzer="word",
stop_words="english", max_features=None)
    fake_news_vectors =
word_vectors.fit_transform(fake_news_data["text"])
    news_vectors2 = TfidfVectorizer(lowercase=True, analyzer="word",
stop_words="english", max_features=None)
    real_news_vectors =
news_vectors2.fit_transform(real_news_data["text"])

    fakeWordsFreq = {}
    realWordsFreq = {}

    for word, index in word_vectors.vocabulary_.items():
        sumOfColumn = fake_news_vectors.getcol(index).sum()
        fakeWordsFreq[word] = sumOfColumn

    for word, index in news_vectors2.vocabulary_.items():
        sumOfCol = real_news_vectors.getcol(index).sum()
        realWordsFreq[word] = sumOfCol

    fake_word_cloud = WordCloud(width=1280, height=720, mode='RGBA',
background_color='white', max_words=3000).fit_words(fakeWordsFreq)
    plt.imshow(fake_word_cloud)
    plt.axis("off")
    plt.show()
    plt.savefig("C:/outputs/fake_news_tags.png", format="png")

    real_word_cloud = WordCloud(width=1280, height=720, mode='RGBA',
background_color='white', max_words=3000).fit_words(realWordsFreq)
    plt.imshow(real_word_cloud)
    plt.axis("off")
    plt.show()
    plt.savefig("C:/outputs/real_news_tags.png", format="png")
```

Figure 6.2: Python code example for the word cloud generation

6.2.2.3 Constraints

The selected dataset(s) will depend on the problem definition. To process larger datasets, where the number of articles exceeds 1 million records, more computational power may be required.

6.2.3 Select Appropriate Data Cleansing Techniques

6.2.3.1 Problem Definition

Presently, many data cleansing techniques exist, but evidence uncovered by researchers suggests that the application of some techniques might not improve classification results. In parts of the experiments conducted by Silva et al. (2020), the authors use various combinations of data cleaning techniques, such as stop word removal and stemming, and found such approaches did not improve the performance of the models, compared to experiments that had no data cleaning techniques applied (Silva et al., 2020).

6.2.3.2 Solution

Before attempting any data cleansing, it would be worthwhile exploring literature specific to the problem domain. Furthermore, using a simpler set of data cleansing techniques, such as stop words removal and special characters removal, can produce great results. Stop words removal is the process of removing words that do not add value to a given sentence. Figure 6.3 and Figure 6.4 show the pseudocode and Python code required to implement stop words removal. Figure 6.5 and Figure 6.6 show the pseudocode and Python code required to implement special characters removal.

```
START
define function remove_stop_words():
        get a list of well-known stop words
        for each column in title and text columns of the articles dataset:
                initialize counter to 0
                for each index and row in the articles dataset:
                        split words in the current column by tokenizing the words
                        only store words which do not appear in the stop words list
                        update the current article’s title or text by storing the filtered word tokens
                        increment counter by 1
END
```

Figure 6.3: Pseudocode example for stop words removal

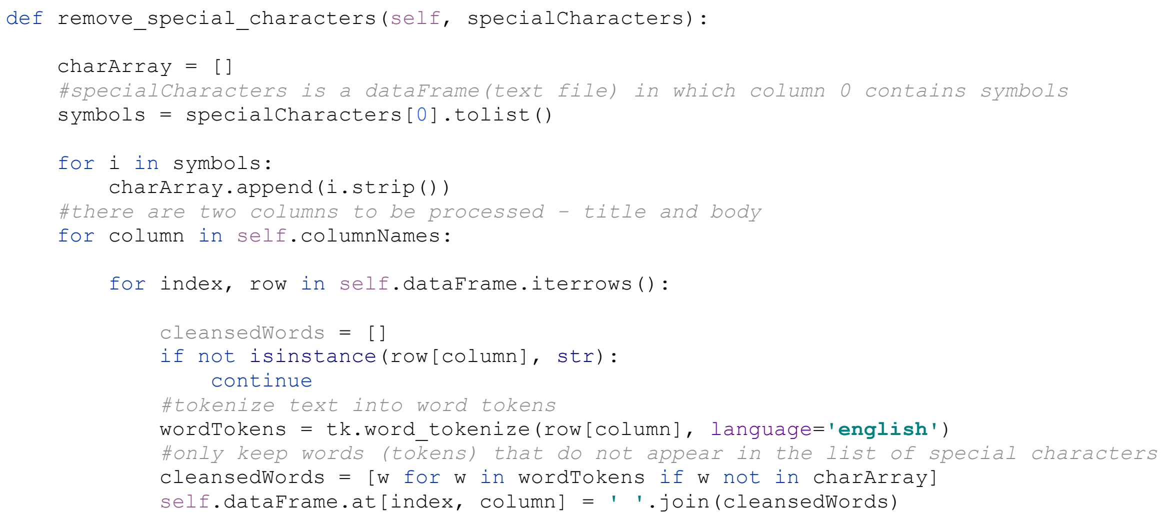
```
def remove_stop_words(self):
    """This function handles the removal of stop words from the corpus. Stop
words are words that do not add value to the body of text"""
    #:cp = ntlk.corpus alias
    #load the pre-populated English stop words list (part of the nltk package)
    stopWords = cp.stopwords.words('english')
    #two columns are loaded into this class, the title and body columns
    for column in self.columnNames:

        row_number = 0
        #dataFrame is the loaded fake and real news dataset.
        for index, row in self.dataFrame.iterrows():
            if not isinstance(row[column], str):
                row_number = row_number + 1
                continue
            #split the text into word tokens
            tokenizedWords = tk.word_tokenize(row[column], language='english',
preserve_line=False)
            #only keep words that do not appear in the stopwords list
            cleansedWords = [w for w in tokenizedWords if w not in stopWords]
            self.dataFrame.at[index, column] = ' '.join(cleansedWords)
            row_number = row_number + 1
```

Figure 6.4: Python code example for stop words removal

```
START
define function remove_special_characters():
        initialize empty list of characters
        get list containing characters to filter against
        for each symbol in the characters list:
                for each column in the article dataset’s column’s title and text:
                        for each index and row in articles set
                                split words in the current row and column of the article set by tokenizing the words
                                only keep words that do not appear in the special characters array
                                update current article row and column text by storing the filtered words
END
```

Figure 6.5: Pseudocode example for special characters removal



```
def remove_special_characters(self, specialCharacters):

    charArray = []
    #specialCharacters is a dataFrame(text file) in which column 0 contains symbols
    symbols = specialCharacters[0].tolist()

    for i in symbols:
        charArray.append(i.strip())
    #there are two columns to be processed - title and body
    for column in self.columnNames:

        for index, row in self.dataFrame.iterrows():

            cleansedWords = []
            if not isinstance(row[column], str):
                continue
            #tokenize text into word tokens
            wordTokens = tk.word_tokenize(row[column], language='english')
            #only keep words (tokens) that do not appear in the list of special characters
            cleansedWords = [w for w in wordTokens if w not in charArray]
            self.dataFrame.at[index, column] = ' '.join(cleansedWords)
```

Figure 6.6: Python code example for special characters removal

6.2.4 Select Features That Best Differentiate Fake from Real News

6.2.4.1 Problem Definition

In the natural language processing field, a wide range of textual features can be extracted from text. Examples include parts of speech tags, number of named entities, punctuation counts, sentence counts, and more. Simplistic, accumulative features can deliver promising classification results, but these features do not account for all the indicators which can differentiate fake news articles from real news articles. Fake news articles are underpinned with stylistic, sentence and article compositions, and readability differences. The exclusion of such features could result in a classification model that is not able to learn from indicators or cues highlighted by other researchers for identifying fake news articles.

6.2.4.2 Solution

An optimal feature set for fake news detection should not solely rely on simplistic, accumulative features like word counts and document lengths. Instead, the feature set should be supplemented with additional features that capture other aspects of fake news, such as the relevance of the article title and body, the readability of articles, and more. Through literature review and experimentation, the following set of indicators are defined:

1. Fake news articles generally contain short article bodies and longer article titles.
2. Fake news articles generally use simpler sentence structures.
3. Fake news article’s generally have headlines which aren’t related to the article body.

To create a feature set that addresses the above indicators, the following calculative features are included, and should be included with any feature sets for text-based fake news detection:

1. **Cosine Similarity**: A calculation that measures the similarity between two pieces of text.
2. **Type-Token Ratio**: A calculation that measures the total number of unique words against the total number of words in a document
3. **Text Readability Metrics**: Gunning Fog Index, Automated Readability Index and Flesch Kincaid Grade Level Index. These are readability metrics that grade the readability of text.
4. **Other lengths**: Average sentence lengths, title and article body word counts, title and body punctuation counts.

The py-readability-metrics Python package can assist in streamlining the process around calculating the readability metrics. The lexical-richness Python library can be used for calculating the Type-Token ratio. To quantify the readability of news articles, Horne & Adali (2017) use the Gunning Fog, SMOG and Flesch-Kincaid Grade readability scores to measure the readability of articles. Several researchers have employed cosine similarity to calculate the similarity between a given article title and text (Ferreira & Vlachos, 2016; Silva et al., 2020; Wu et al., 2017). Figures 6.7 and 6.8 show code and pseudocode examples of a function that calculates the type-token ratio and unique word count for each article in a dataset. Figures 6.9 and 6.10 show examples of pseudocode and code for calculating text readability scores.

```
START
define function calculate_type_token_and_unique_words():
      add a column to the feature set to keep type-token ratio scores
      add a column to the feature set to keep unique word count scores
      initialize row counter to 0
      for each item and row in the articles set:
            get the type-token ratio score for the current article’s
            text
            get the unique words score for the current article’s text
            add the type-token ratio score to the feature set
            add the unique words score to the feature set
            increment row counter by 1
     return feature set
END
```

Figure 6.7: Pseudocode example for type-token ratio and unique word count calculations

```
def calculate_ttr_and_unique_words(self, featureDataFrame):
    print("Calculating Type-Token Ratio and Unique Words: ")
    featureDataFrame["tt_ratio"] = float(0.0)
    featureDataFrame["uq_words"] = 0
    rowCounter = 0

    for i, row in self.dataFrame.iterrows():
        #get a reference to the text (article body)
        content = row["text"]
        #instantiate the lexical richness object, pass the text (article
body)
        lex = lr.LexicalRichness(use_TextBlob=True, text=content)
        #calculate the type-token ratio score... round to 2 decimal
points
        ttr_score = round(float(lex.ttr), 2)
        featureDataFrame.at[rowCounter, "tt_ratio"] =
str(float(ttr_score))
        #the lexical richness library can also be used to calculate the
number of unique words in a given text.
        featureDataFrame.at[rowCounter, "uq_words"] =
pd.to_numeric(lex.terms)
        rowCounter = rowCounter + 1
        print("TTR Ratio for Article " + str(rowCounter) + " is: " +
str(ttr_score))
        print("Unique Words for Article " +str(rowCounter) + " is: " +
str(lex.terms))
```

Figure 6.8: Python code example to calculate the type-token ratio and unique word counts.

```
START
define function get_readability_score():
    add automated readability index feature to the features dataset
    add gunning fog index feature to the features dataset
    add flesch Kincaid grade level index to the features dataset
    set counter to 0
    for each index and article row in articles:
        initialize metrics to Readability(row[‘text’]) instance
        calculate automated readability index for an article’s text
        calculate gunning fog index score for an article’s text
        calculate flesch Kincaid grade level score for an article’s text
        add calculated automated readability score to the features set
        add calculated gunning fog index score to the features set
        add calculated flesch Kincaid score to the features set
        increment counter by 1
    return features dataset
END
```

Figure 6.9: Pseudocode example to calculate text readability scores

```
def get_readability_score(self, featureDataFrame):

    #initialize a few variables
    ari =  "text_ari"
    gfIndex = "text_gf"
    fkIndex = "text_fkg"
    featureDataFrame[ari] = float(0.0) # automated readability index
    featureDataFrame[gfIndex] = float(0.0) # fog index
    featureDataFrame[fkIndex] = float(0.0) # Flesch Kincaid Grade Level index

    counter = 0
    #iterate through all articles in the dataset
    for index, row in self.dataFrame.iterrows():
        #create an instance of the Readability metrics
        readabilityMetrics = Readability(row["text"])

        #get the automated readability index score
        featureDataFrame.at[counter, ari] = readabilityMetrics.ari().score
        #get the gunning fog index score
        featureDataFrame.at[counter, gfIndex] =
readabilityMetrics.gunning_fog().score
        #get the flesch-Kincaid score
        featureDataFrame.at[counter, fkIndex] =
readabilityMetrics.flesch_kincaid().score
        counter = counter + 1
```

Figure 6.10: Python code example for calculating text readability metrics

6.2.4.3 Other Considerations

Due to the design of the py-lexical-richness library, to calculate readability scores for any piece of text, the text must contain a minimum of 100 words. (DiMascio, 2019).

6.2.5 Consider Various Classification Models for Fake News Detection

6.2.5.1 Problem Definition

Both traditional machine learning and deep learning realms have models which can be useful for the task of fake news detection. The experimentation in chapter 5 shows that results are variable, depending on data cleansing, feature extraction processes and the selected model. Therefore, the process of simply selecting a single model may not be sufficient to determine the optimal classification model. Furthermore, some machine learning or deep learning models have limitations that may not be suitable for all classification problems. Presently, there exists a wide variety of traditional machine learning algorithms, vanilla deep learning models, and hybrid deep learning models for the task of fake news detection. In the case of recurrent neural networks, Agarwal et al. (2020) highlight the high computation costs exhibited as a result of large texts.

6.2.5.2 Solution

For deep learning approaches that can handle explicit features and latent features present in the body of text, hybrid deep learning architectures are recommended. For classification problems where large volumes of texts are supplied, the CNN-DNN hybrid model is recommended owing to its speed, simplicity, and lower computation costs. Keras is the recommended Python library to construct such a hybrid model.

For machine learning approaches, the Support Vector Machine algorithm is recommended, owing to the superior results it obtains compared to other machine learning algorithms. Figure 6.11 shows the pseudocode required to build the hybrid CNN model. Figures 6.12 to 6.17 show the Python code required to build the hybrid CNN model.

```
START
Define a model for explicit text features:
get implicit features data from file
create Keras explicit model by setting the Input layer as the first layer. Use the implicit features as input
add a dense layer to the Keras model
add a dropout layer to the Keras model
add a dense layer to the Keras model
add a dropout layer to the Keras model
Define a model for implicit text features:
create Keras implicit model by adding the Input layer as the first layer.
Add embedding layer to the Keras implicit model
add convolutional neural network layer to the Keras implicit model
add a max-pooling layer to Keras implicit model
add dropout layer to the Keras implicit model
add flatten layer to the Keras implicit model
Merge the two models (concatenate):
merge the explicit and implicit Keras models through concatenation
create final Keras layer, using the merged model
add dense layer to the final Keras layer
add dropout layer to the final Keras layer
add the output layer (dense layer) to the final Keras layer
Compile the hybrid model:
create final Keras model by adding the explicit and implicit inputs and final output layer
compile the final Keras model by specifying the metrics and optimizer to use.
END
```

Figure 6.11: Pseudocode example for the construction of a hybrid CNN model

Define Model for Explicit Text Features:

```
# import the layers module from keras library:
import tensorflow.keras.layers as l
explicitModelInput = l.Input(name=”InputLayer1”,
shape=(explicitModelFeatures.shape[1],))
explicitModel = l.Dense(name=”DenseLayer1”, units=192,
                activation=tf.keras.activations.relu,
                bias_initializer=tf.keras.initializers.Zeros(),
                kernel_initializer=tf.keras.initializers.GlorotUniform())
               (explicitModelInput)
explicitModel = l.Dropout(name=”Dropout1”, rate=0.5)(explicitModel)
explicitModel = l.Dense(name=”DenseLayer2”, units=64,
activation=tf.keras.activations.relu)(explicitModel)
```

Figure 6.12: Python code example for hybrid CNN

Define Model for Implicit Text Features:

```
# import the layers module from Keras library:
import tensorflow.keras.layers as l

embedding_inputs = l.Input(name=”InputLayer2”, shape=(maximum_words,))
# Define embedding layer – use word embeddings as initial weights
embedding_layer = Embedding(len(unique_words_array), 300,
                  trainable=False, weights=[document_word_embeddings],
                  input_length=len(unique_words_array),name=”EmbeddingOne”)

implicitModel = embedding_layer(embedding_inputs)
```

Figure 6.13: Python code example for hybrid CNN (continued)

```
# Declare and configure Conv1D cell
implicitModel = Conv1D(name=”Conv1Done”, filters=5, kernel_size=3,
                activation=’relu’,strides=2,
                kernel_initializer=tf.keras.initializers.GlorotUniform(),
                padding=’same’,
                bias_initializer=tf.keras.initializers.Zeros())
               (implicitModel)
# Add Max-Pooling layer
implicitModel = MaxPool1D(name=”MaxPoolOne”, pool_size=3)(implicitModel)
# Regularization technique: Add dropout layer
implicitModel = Dropout(name=”DropoutLayer3”, rate=0.5)(implicitModel)
implicitModel = Flatten()(implicitModel)
```

Figure 6.14: Python code example for hybrid CNN (continued)

Merge (concatenate) the two models:

```
merged = Concatenate(name=”Merge”)([explicitModel, implicitModel])
# add final layers
finalLayer = Dense(units=128,activation=tf.keras.activations.relu)(merged)
finalLayer = Dropout(rate=0.4)(finalLayer)

# output layer - use the sigmoid function for binary classification
problems
finalLayer =
       Dense(units=1,activation=tf.keras.activations.sigmoid)(finalLayer)
“””Merge both implicit and explicit inputs - one output for the predicted
class”””
finalModel = Model(inputs=[embedding_inputs, explicitModelInput],
       outputs=[finalLayer], name=”CNN_DNN”)
# OPTIONAL: Log training results to CSV file for further analysis
csvFile = CSVLogger(filename=”../data/cnn_dnn_2_1.csv”, separator=’;’,
       append=False)
```

Figure 6.15: Python code example for hybrid CNN (continued)

Compile the Hybrid Model and Perform Training:

```
# Compile the model - add performance metrics
finalModel.compile(optimizer=tf.keras.optimizers.Adam(),
   metrics=[‘accuracy’,
           tf.keras.metrics.TrueNegatives(),
           tf.keras.metrics.TruePositives(),
           tf.keras.metrics.FalseNegatives(),
           tf.keras.metrics.FalsePositives(),
           tf.keras.metrics.AUC(),
           tf.keras.metrics.Recall(),
           tf.keras.metrics.Precision(),
           ],
   loss=tf.keras.losses.BinaryCrossentropy())
plot_model(finalModel, to_file=”../data/merged_model_final13_2.png”,
   show_shapes=True, show_layer_names=True)
```

Figure 6.16: Python code example for hybrid CNN (continued)

```
finalModel.summary()
finalModel.fit(x=[np.asarray(imp_x_train), np.asarray(explicit_x_train)],
               y=np.asarray(imp_y_train),
               callbacks=[csvFile],
               validation_data=([np.asarray(imp_x_test),
                   np.asarray(explicit_x_test)], np.asarray(imp_y_test)),
               epochs=20)
```

Figure 6.17: Python code example for hybrid CNN (continued)

6.2.5.3 Constraints

The selected models will be dependent on the dataset, sample sizes, and the problem definition.

6.2.6 Determine Optimal Configuration for Selected Classification Model(s)

6.2.6.1 Problem Definition

The effectiveness of machine and deep learning models are greatly influenced by the supplied hyperparameters. Although researchers mention the use of specific parameters and values, there is no guarantee such metrics can produce optimal performance results, relative to the classification problem. Though several Python libraries, such as Keras-Tuner and SciKit-Learn, provide APIs which assist in determining the optimal model configuration, the use of such APIs can be dependent on other factors, such as available system resources. The two most used selection processes are Grid Search and Random Search. Both techniques operate differently, and their usage depends on expectations. Given a set of hyperparameter values, Random Search randomly selects a combination of hyperparameters, whilst Grid Search selects and evaluates all hyperparameter values (Brownlee, Hyperparameter Optimization With Random Search and Grid Search, 2020). Cuevas et al. (2020) state deep learning algorithms have hyperparameters that can influence memory usage and execution cost. Such parameters are dependent on the task and selected dataset (Cuevas et al., 2020).

6.2.6.2 Solution

Both Keras-Tuner and SciKit-Learn libraries provide systematic approaches to determine a given model's best configuration. For machine learning workflows, the APIs provided by Sci-Kit Learn is recommended. For deep learning workflows, the APIs provided by Keras-Tuner are recommended. Grid Search could be used in cases where hardware limitations are not an issue, whereas Random Search could be used in cases where hardware limitations would otherwise cause much longer processing times.

```
define function grid_search_param_selection(model, parameters, features, labels):
      initialize grid search using the declared machine learning classifier and features
fit the labels into the grid search instance
get the best parameters scores from the grid search instance
return best parameters
END
```

Figure 6.18: Pseudocode example to implement the GridSearch technique

```
START
def perform_grid_search_param_selection(estimator, param_dictionary,
x_dataset, y_labels):
    """This function performs the Grid Search technique, given a set of
parameters
    Parameters
    --------------
    estimator: A pre-built machine learning SciKit-Learn model
    param_dictionary: A JSON style object containing hyper-parameter and
values to test
    x_dataset: The dataset to fit model
    y_labels: A collection of labels for each sample in the x_dataset
    """
    #define the Grid Search Model
    grid_searcher = ms.GridSearchCV(estimator=estimator,
param_grid=param_dictionary, scoring="accuracy", n_jobs=4, cv=3,
return_train_score=True)
    #fit the model with features and labels data - may take a while
depending on model type and data sizes
    grid_searcher.fit(x_dataset, y_labels)
    #get the best parameter configurations
    return grid_searcher.best_score_
```

Figure 6.19: Python code example for implementing grid search technique

```
xgboost_params = [
     {'n_estimators': [7, 10, 50, 100],
      'max_depth': [1, 3, 5, 7]
      }]
```

Figure 6.20: Example of setting up hyperparameter search space for the grid search.

```
START
define function build_model(hp):
      create Keras model by adding Input layer
      add embedding layer to the Keras model
      add convolution neural network layer to the Keras model. Use Keras-
Tuner to set up parameter test values.
      add a dense layer to the Keras model. Use Keras-Tuner to set up
parameter test values.
      add dropout layer to the Keras model. Use Keras-Tuner to set up
parameter test values.
      add output layer to the Keras model
      create the final model by using inputs (Keras model) and output layer
return final model
```

Figure 6.21: Pseudocode example for setting up a neural network model, for hyperparameter selection

```
initialize random search instance using the function to build the Keras
model. Specify trials and maximum executions per trial.
execute random search
get the best random search results
get results summary from the random search
END
```

Figure 6.22: Pseudocode for performing hyperparameter tuning, after building a test neural network model

```
# create a function that accepts an object which is used to define hyper-parameter search
# bands for parameters of a given model
def test_model(hp):
    #The purpose of this function is to build a model that Keras-Tuner uses to determine
the optimal configuration
    cnn_input_model = Input(shape=(encoded_documents.shape[1],), name="InputLayer")

    model_one = Embedding(len(unique_words_array),
                          300,
                          input_length=maximum_words,
                          trainable=False,
                          weights=[document_word_embeddings],
                          name="EmbeddingLayer")(cnn_input_model)

    #The hp.Choice class is used to define the values used for hyperparameter selection
    model_one = Conv1D(filters=hp.Choice(
                          name="conv_layer_one_filters",
                          values=[70, 90, 110]),
                       kernel_size=hp.Choice(
                          name="conv_layer_one_kernel",
                          values=[2, 3, 4]),
                       activation='relu',
                       strides=hp.Choice(
                          name="conv_layer_one_strides",
                          values=[2, 3, 4]),
                       kernel_initializer=k.initializers.GlorotNormal(),
                       bias_initializer=k.initializers.Zeros(),
                       name="Conv1DOne")(model_one)

   #"NOTE: Other layers have been omitted. These layers would also follow the same process
when defining values to test against
    output_layer = Dense(name="DenseLayerFinal",
units=hp.Choice(
name="dense_final_layer_units",
                                    values=[16, 32, 64]))
```

Figure 6.23: Python code example for constructing a test neural network model for hyperparameter tuning

```
model_training_results = kt.RandomSearch(test_model,
                                         objective='val_accuracy',
                                         directory='CNN_Implicit_2',
                                         project_name='cnn_implicit_2',
                                         max_trials=3,
                                         executions_per_trial=2)

#Similiar to the fit() method, this method performs the search, using provided data
model_training_results.search(np.asarray(x_train),
```

Figure 6.24: Python code example for executing hyperparameter tuning of a neural network.

6.2.6.3 Constraints

Available computational resources, such as memory and CPU, could be a factor that influences the chosen hyperparameter selection method.

6.2.6.4 Other Considerations

The SciKit-Learn implementation of Grid Search has two parameters that could be useful, in terms of performance: ‘n_jobs’ and ‘cv’. The n_jobs parameter determines the number of jobs to run concurrently whilst the ‘cv’ parameter determines the number of folds to perform cross-validation (SciKit Learn, 2012).

The Random Search implementation of the Keras-Tuner library provides two parameters that control the number of models to be built, and the number of tests performed on each model, namely, the ‘max_trials’ and ‘executions_per_trial’ parameters (Keras-Tuner, 2019).

6.2.7 Determine Applicable Metrics to Measure Performance of the Classification Model

6.2.7.1 Problem Definition

Several performance metrics exist which can measure the effectiveness of machine learning and deep learning models. Each performance metric reports on a different aspect of the overall model’s performance. Using a single metric, such as the accuracy of the model, may not be enough to truly quantify the overall performance of a classification model. This is evident in cases where there could be data imbalances between several classes. A poor selection in performance metrics may create misleading results, which in turn, could create a model that does poorly when evaluating unseen data.

6.2.7.2 Solution

Cross-validation techniques should be used to give a better view of the model’s classification performance. In cases where a data imbalance exists in several classes, the stratified k-fold cross-validation technique should be used. Pathak (2019) uses the stratified k-fold cross-validation over 5 folds, owing to the data being imbalanced across several classes. In cases where there is an approximate or equal balance in samples across all classes, the k-fold cross-validation should be used. Most authors employ cross-validation over a set number of folds (Elhadad et al., 2020; Hassan et al., 2017; Horne & Adali, 2017; J. Zhang et al., 2016). In addition to the cross-validation data (stratified or k-fold), the following performance metrics should be included:

1. **Confusion Matrix Results:** Tallies on the number of true positives, true negatives, false positives, and false negatives identified for a given classification problem.
    a. **True Positives:** The number of positive (truthful) samples correctly identified as positive samples.
    b. **True Negatives:** The number of negative (untruthful) samples correctly identified as negative samples.
    c. **False Positives:** The number of negative samples identified as positive samples.
    d. **False Negatives:** The number of positive samples identified as negative samples.
2. **Accuracy:** This metric measures the overall number of correctly labelled samples (positive and negative) samples, across all samples in the dataset. The mathematical expression can be denoted as:
$$Accuracy = \frac{TP + TN}{(TP + FP + TN + FN)}$$
3. **Precision:** This metric measures a model's ability to identify positive (true) samples. Mathematically, this metric can be expressed as:
$$Precision = \frac{TP}{(TP + FP)}$$
4. **Recall:** This metric measures a model's ability to correctly label positive samples amongst all known positive samples. Mathematically, this metric can be expressed as:
$$Recall = \frac{TP}{(TP + FN)}$$
5. **F-Measure:** This metric is the average between Precision and Recall (Abdullah-All-Tanvir, Mahir, Akhter, & Huq, 2019). Mathematically, this metric can be expressed as:
$$F - Measure = \frac{2\ x\ (Precision\ x\ Recall)}{Precision + Recall}$$
6. **Receiving Operator Characteristics:** This metric measures the trend between the True Positive Rate (TPR) and False Positive Rate (FPR) at different thresholds (Nasir, Khan, & Varlamis, 2021). The True Positive Rate (TPR) is calculated by summing up the total true positives, divided by the sum

of false positives and true negatives. The TPR can be expressed as (Tchakounté & Hayata, 2016):

$$TPR = \frac{FP}{TP + FN}$$

The False Positive Rate (FPR) is calculated by summing up all false positive samples, divided by the sum of false positive and true negative samples. The FPR can be experessed as (Tchakounté & Hayata, 2016):

$$FPR = \frac{FP}{FP + TN}$$

```
START
   define function perform_k_fold_cross_validation(num_folds, classifier,
   features, labels):
      print name of classification model
      initialize counter as 1
      using classification model, features, and labels as parameters, get
   cross-validation results scores
   for each score in scores:
      print current fold number and the score obtained
      increment counter by 1
END
```

Figure 6.25: Pseudocode example for K-Fold cross validation

```
def perform_kfold_cross_validation(self, numFolds, classifier, x_dataset, y_dataset):
    """This function performs cross-validation for a given classification model.
    Parameters
    -------------
    numFolds: The number of folds for the cross-validation
    classifier: The SciKit-Learn machine learning model
    x_dataset: The feature set to be used for cross-validation (features).
    y_dataset: labels for each sample in the provided feature set"""
    crossValScore = cross_val_score(classifier, x_dataset, y_dataset, cv=numFolds,
n_jobs=4)
    print("K-Fold Cross Validation Complete for "+classifier.__class__.__name__)

    counter = 1
    for i in crossValScore:
        #Output score for each fold
        print("Fold " + str(counter) + " Cross Validation Score: " + str(i))
        counter = counter + 1
```

Figure 6.26: Python code example for implementing k-fold cross-validation

6.2.7.3 Constraints

The selected metrics will depend on the problem definition; for classification problems, a set of applicable metrics exist. For regression problems, other applicable metrics also exist.

6.3 Conclusion

In this chapter, a comprehensive set of guidelines is presented. Such guidelines are supported by the work of other researchers in the natural language processing field, and data obtained in the experimentation phase of this study. In addition, code examples and references to useful tooling required to build such systems are included. These guidelines aim to serve as a guide for developers or researchers studying or building systems for combatting online misinformation. Part of what makes fake news detection a complex problem to solve can be attributed to many reasons; presently, due to the many definitions of what is deemed as fake news, there is not a single, accepted definition of fake news. In addition, there is no single fake news dataset – numerous datasets for the classification task vary in terms of article lengths, the age of the dataset, the topics covered in each of the datasets, and the languages (locality) of the datasets. To add to the complexity, several machine learning and deep learning approaches are available, however, some approaches may not be suitable, depending on constraints such as system resources and the problem at hand. In the experimentation covered in Chapter 5, text summarization was employed to lower the high computation costs exhibited by recurrent neural networks. The guidelines presented in this chapter provide insight into fake news detection which can be used in developing systems aimed at combatting this problem.

To conclude, the guidelines presented in this chapter address the primary research objective of the project:

> *Develop guidelines for the use of machine learning to identify text-based fake news*

Chapter 7 provides an overview of all the work carried out in this project and how the work collectively addresses the problem statement and research objectives. Additionally, Chapter 7 also discusses the contributions this work makes and lists some limitations which can be overcome in future works.

7 Chapter 7 - Conclusion

This chapter aims to conclude the study undertaken in this project by presenting a summary of the research objectives and how they were answered, at various phases of the project. A summary of chapters is presented and the common thread between the chapters is also shown. Limitations and future work related to this study are also addressed.

7.1 Introduction

The spread of misinformation can lead to many consequences from a social and economic perspective. What makes fake news a complex problem to solve is the varying definitions of what is deemed fake news, and the multitude of datasets that span across various text lengths. The rapid spread of online fake news makes it difficult for users to verify the credibility of articles that circulate on the internet. The lack of fact-checking on social media platforms further adds to the difficulty in preventing the spread of misinformation. Researchers in the fake news detection community have shown successes in using machine learning and deep learning approaches to tackle the problem of online misinformation. Through the literature of other researchers, a set of indicators that can differentiate fake news articles from real news articles is defined. Using the set of indicators, machine learning and deep learning approaches are employed to validate the effectiveness of automated fake news detection and the applicability of the defined indicators. Through experimentation, the performance results obtained show that such approaches can prove to be effective in combatting online misinformation.

This chapter presents a summary of all chapters in this study, followed by the problem statement and how this statement is addressed in this study. A section that describes the contributions this study makes, and the limitations of this research project, is also included in this chapter. Finally, the chapter concludes by presenting suggestions that could be used in future work related to this study.

7.2 Research Objectives

The primary objective of this study was to *develop guidelines for the use of machine learning to identify fake news.* To address this objective, a set of guidelines are presented, which can be used in developing computational systems and tooling that tackles online misinformation. To support the credibility and validity of the guidelines, information uncovered from examining the literature and information uncovered from experimentation are considered.

Following the primary objective, the study defines three secondary objectives. These are:

1. *Identify the motives behind the propagation of fake news.*
2. *Identify appropriate indicators, from literature, which could suggest an article is fake news.*
3. *Validate the identified indicators through the application of selected machine learning algorithms.*

The purpose of the first secondary objective was to understand why fake news is a problem that should be investigated. The problem of online misinformation, and the complexities around addressing the issue, are highlighted in literature chapters 2 and 3, and further revisited in the experimentation chapter. Using information uncovered in secondary objective 1, secondary objective 2 aims to define a list of indicators, or cues, which could differentiate fake news articles from real news articles. The premise for such guidelines is based on observations made by other researchers, and literary information on differences in fake and real online news. Using the guidelines defined through provisions of secondary objective 2, the effectiveness of these are indicators are evaluated through existing machine learning and deep learning approaches. In a machine learning workflow, the guidelines translate to useful features that can best differentiate fake articles from real news articles. The study selects several machine learning and deep learning models, based on the information and the results obtained by other researchers pertaining to the selected models. To quantify the effectiveness of the identifiers, a host of performance-related metrics are selected. Chapter 5 presents the results obtained, a discussion of the results, and how such results align with existing literature. Suggestions around improving the obtained results are included in the experimentation chapter.

The information uncovered through secondary objective 1, together with the indicators of fake news articles defined by secondary objective 2, and the results obtained by performing automated fake news detection using machine learning approaches and the defined indicators, are used as the supporting logic in the development of the guidelines for fake news detection proposed in this project.

7.3 Problem Statement

The problem statement defined in this study is *"Due to the rapid spread of online misinformation, in the form of fake news, manual approaches employed to differentiate fake news from real news articles, are time consuming and inadequate, given the rapid dissemination of online news".* Through findings uncovered in the literature review, it is shown that online news readers' stances on certain topics can change, depending on the information disseminated online. Furthermore, cases in which fake news has caused economics-related shifts have been outlined in the literature. To address the problem, indicators that can differentiate false from real news articles are defined. The premise supporting the indicators are based on findings and observations made by other researchers in the fake news detection community. Following the definition of indicators, a selection of machine learning algorithms and deep learning models are examined. The results obtained show the problem of misinformation can be averted, using computational remedies.

7.4 Summary of Chapters

Chapter 1 provides a brief overview of the research project by presenting literature about the problem. A problem definition, a thesis statement, research objectives, ethical considerations, and the overall research process and design are defined. The chapter concludes by presenting an overview of the proposed chapters for the study, and how the structure of the study is designed to answer the research question.

Chapter 2 expands on the foundations laid by chapter 1, by covering online fake news in detail. Through examining existing literature, this chapter defines fake news from a global and South African context. This chapter outlines the consequences of fake news, the benefits for the creators of fake news content, the current efforts around detecting fake news, and the complexities around detecting fake news. This chapter is necessary to understand fake news, and why it is a problem worth solving.

Chapter 3 continues the literature review, this time examining the best machine and deep learning approaches used to detect fake news. This chapter touches on the available technologies, and how these approaches were applied by other researchers. Performance results and notable performance-related issues are also mentioned in this chapter. This chapter serves as the foundation needed to carry out experimentation for fake news detection.

Chapter 4 presents an overview of the research process and design undertaken. The selected methods are defined with a clear connection to the secondary objectives defined in chapter 1. The chapter concludes by presenting an overview as to how the research methods employed support the primary research objective.

Chapter 5 covers all experimentation undertaken in this study. Using the identified indicators, and several machine learning and deep learning models, the concept of fake news detection is examined over two datasets. The results of the experimentation are discussed and compared with existing literature.

Chapter 6 presents a set of guidelines for the task of fake news detection. For each guideline, a brief description of the problem, a solution and in some cases, code snippets are provided. This chapter aims to present a set of guidelines that can be used by other researchers and developers, to aid in the task of misinformation detection.

Chapter 7 reflects on all the work undertaken in this study and concludes the project.

7.5 Contributions of the Study

Through the process of literature review, this study shows that fake news detection, using machine learning and deep learning approaches, is possible, using indicators that best separate fake news from real news articles. The effectiveness of these indicators, and the applicability of computational approaches to the problem, are shown in the promising results obtained through experimentation in Chapter 5. Part of the machine learning related experimental data and findings have been published, in the form of a conference paper. Details about the conference paper can be found in Appendix A.

In addition, the study presents a set of guidelines that could be used for future work in building systems for automated fake news detection. The guidelines aim to provide a point of reference that could be considered at various stages of an automated fake news detection development workflow. The guidelines are built using information uncovered in the literature, and experimentation carried out in chapter 5, as the founding logic. To illustrate the practicality of the guidelines, some concepts highlighted in the guidelines are presented in the form of source code.

7.6 Limitations and Future Work

This study explores the applicability of indicators, or cues, which could be used in differentiating fake news articles from real news articles. Through experimentation covered in chapter 5, the applicability of such indicators is examined using several machine learning and deep learning approaches. The high accuracy results show that models aimed at detecting false news content, using the selected indicators, are viable.

7.6.1 Limitations

Though the experiments show that promising results are attainable using the approaches described in chapter 5, there are several limitations, which could be improved on in future studies:

1. **Only textual article headlines and bodies are considered:** In a real-world sense, it is common for articles to include images and videos which can be fabricated for the intent of spreading fake news. In this study, only textual article headlines and article bodies are examined.
2. **User metrics and news source information are not examined:** It is common for online news articles to include user information, such as user comments, number of likes or dislikes, and shares. Additionally, information relating to the source of the online article, such as the site's online ranking and age, are not examined. As highlighted in chapter 3, researchers have included such metrics in their fake news detection and have shown that such metrics can influence the overall accuracy of developed models.
3. **Hardware-related limitations:** The experiments were performed offline, on a personal computer. Deep learning architectures, such as Recurrent Neural Networks, can be expensive to run on systems with limited computational resources. In chapter 5, text summarization is used to reduce computational costs exhibited with the selected Recurrent Neural Network.

7.6.2 Future Work

Future work in fake news detection could include a framework that classifies articles using features that consider an article's images, videos, authors, user interaction, and the credibility of the source. Frontend tools, which could be used by online news readers, could be developed and tested in a focus group. Additionally, the future study could explore the use of cloud computing platforms, such as Google Cloud, Amazon

AWS, or Microsoft Azure, for the development and implementation of user-facing tools and systems. Cloud computing platforms have virtualized solutions built for Machine Learning and Deep Learning workflows, which could be beneficial for solutions that require a large amount of computational power.

7.7 Conclusion

This project aimed to present a set of guidelines that can be used by other developers and researchers looking to explore fake news detection using machine learning approaches. Some difficulties in this field include the complex nature of language, the variances in available datasets, and the varying definitions of what is perceived as fake news. Through examining the work of other researchers, a set of indicators are devised, which serve their purpose in the experimentation phase of the project. The study also shows automated fake news detection is possible, using existing machine learning and deep learning models. Though there is still room for improvement, a set of guidelines has been presented and promising experimentation results were obtained. These guidelines, alongside the growing body of research and development in this field, will hopefully prove to be a useful tool in the fight against fake news.

References

Abdelminaam, D. S., Ismail, F. H., Taha, M., Taha, A., Houssein, E. H., & Nabil, A. (2021, February 9). CoAID-DEEP: An Optimized Intelligent Framework for Automated Detecting COVID-19 Misleading Information on Twitter. *IEEE Access, 9*, 27840 - 27867. doi:10.1109/ACCESS.2021.3058066

Africa Check. (2013, January 13). *Africa Check funding since 2012 | Our funders and expenditure*. Retrieved July 27, 2019, from Africa Check: https://africacheck.org/about-us/how-we-are-funded/

Agarwal, A., Mittal, M., & Goyal, L. M. (2020). Fake News Detection Using a Blend of Neural Networks. *SN Computer Science (2020)*, 1 - 9. doi:https://doi.org/10.1007/s42979-020-00165-4

Ahmed, H., Traore, I., & Saad, S. (2017). Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. *ISDDC: International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*, 127 - 138.

Ahmed, Z., Amizadeh, S., Bilenko, M., Carr, R., Chin, W.-S., Dekel, Y., . . . Zhu, Y. (2019). Machine Learning at Microsoft with ML.NET. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2448–2458). arXiv. doi:10.1145/3292500.3330667

Al-Ash, H. S., & Wibowo, W. C. (2018). Fake News Identification Characteristics Using Named Entity Recognition and Phrase Detection. *Proceedings of 2018 10th International Conference on Information Technology and Electrical Engineering*, (pp. 1 - 6).

Allcott, H., & Gentzkov, M. (2017). Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives*, 211 - 236.

Alom, M. Z., Moody, A. T., Maruyama, N., Van Essen, B. C., & Taha, T. M. (2018). Effective Quantization Approaches for Recurrent Neural Networks. *arXiv*, 1 - 8. doi:arXiv:1802.02615

Anderson, J., & Rainie, L. (2017, October 19). *The Future of Truth and Misinformation Online*. Retrieved March 15, 2018, from Pew Research Center:

http://www.pewinternet.org/2017/10/19/the-future-of-truth-and-misinformation-online/

Bahad, P., Saxena, P., & Kamal, R. (2019). Fake News Detection using Bi-Directional LSTM-Recurrent Neural Network. *Procedia Computer Science*, 74 - 82.

Banik, S. (2020, November 20). *COVID19 Fake News Dataset.* doi:10.5281/zenodo.4282522

Bank, S. (2020, December 24). *graphviz PyPI*. Retrieved May 12, 2021, from pypi: https://pypi.org/project/graphviz/

Batchelor, O. (2017). Getting out the truth: the role of libraries in the fight against fake news. *Reference Services Revieq*, 143 - 148.

Bilal, D., & Huang, L.-M. (2019). Readability and word complexity of SERPs snippets and web pages on children's search queries: Google vs Bing. *Aslib Journal of Information Management, 71*(2), 241 - 259. doi:https://doi.org/10.1108/AJIM-05-2018-0124

Bisaillon, C. (2020, April). *Fake and Real News Dataset | Kaggle*. Retrieved from Kaggle: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

Bisaillon, C. (2020, March 03). *ISOT Fake News Dataset.* Retrieved July 08, 2021, from Kaggle: https://www.uvic.ca/engineering/ece/isot/assets/docs/ISOT_Fake_News_Dataset_ReadMe.pdf

Brownlee, J. (2014, March 21). *Classification Accuracy is Not Enough: More Performance Measures You Can Use*. Retrieved August 04, 2019, from Machine Learning Mastery: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/

Brownlee, J. (2020, January 13). *How to Fix k-Fold Cross-Validation for Imbalanced Classification*. Retrieved July 02, 2021, from Machine Learning Mastery: https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/

Brownlee, J. (2020, September 14). *Hyperparameter Optimization With Random Search and Grid Search*. Retrieved May 01, 2021, from Machine Learning

Mastery: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

Burfoot, C., & Baldwin, T. (2009). Automatic Satire Detection: Are You Having a Laugh? *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, (pp. 161–164). Retrieved from https://aclanthology.org/P09-2041

Campan, A., Cuzzocrea, A., & Truta, M. T. (2017). Fighting Fake News Spread in Online Social Networks: Actual Trends and Future Research Directions. *2017 IEEE Interational Conference on Big Data.*

Chen, B., Chen, B., Gao, D., Chen, Q., Huo, C., Meng, X., . . . Zhou, Y. (2021, January 18). Transformer-based Language Model Fine-Tuning Methods for COVID-19 Fake News Detection. arXiv. Retrieved from https://arxiv.org/abs/2101.05509

Chen, Y., Conroy, N. J., & Rubin, V. L. (2016). News in an Online World: The Need for an "Automatic Crap Detector". *Proceedings of the Association for Information Science and Technlology*, 1-4. doi:https://doi.org/10.1002/pra2.2015.145052010081

Chollet, F. (2019, March 01). *The Functional API*. Retrieved from Keras: https://keras.io/guides/functional_api/

Chollet, F., Rahman, f., Lee, T., de Marmiesse, G., Zabluda, O., Zhu, Q. S., . . . Vijay, P. (2020, June 18). *GitHub - keras-team/keras: Deep Learning for humans*. Retrieved May 01, 2021, from Github: https://github.com/keras-team/keras

Choudary, A., & Arora, A. (2020). Linguistic feature based learning model for fake news detection and classification. *Expert Systems With Applications, Volume 169*, 1 - 15. doi:https://doi.org/10.1016/j.eswa.2020.114171

Choudhary, A., & Arora, A. (2020). Linguistic feature based learning model for fake news detection and classification. *Expert Systems With Applications*, 1 - 15.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modelling. *arXiv*, 1 - 9. Retrieved July 16, 2021, from https://arxiv.org/abs/1412.3555

Clifford, C. (2019, July 22). *German national Michael Duerr owns 0.5% – not 57.5% – of South Africa’s Reserve Bank* | *Africa Check*. Retrieved July 27, 2019, from

Africa Check: https://africacheck.org/reports/german-national-michael-duerr-owns-0-5-not-57-5-of-south-africas-reserve-bank/

Comrie, S. (2017, January 24). *EXCLUSIVE: The ANC's R50m election 'black ops'*. Retrieved June 17, 2019, from News24: https://www.news24.com/SouthAfrica/News/exclusive-the-ancs-r50m-election-black-ops-20170124

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2018, July 8). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. 1 - 12. arXiv. Retrieved October 11, 2021, from https://arxiv.org/abs/1705.02364

Dangeti, P. (2017). *Statistics for Machine Learning.* Birmingham: Packt Publishing Ltd. Retrieved July 02, 2021, from https://www.packtpub.com/product/statistics-for-machine-learning/9781788295758

Deng, X., Liu, Q., Deng, Y., & Mahedevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 250 - 261.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiV*, 1 - 16.

DiMascio, C. M. (2019). *py-readability-metrics — py-readability-metrics 1.4.4 documentation*. Retrieved May 01, 2021, from py-readability-metrics: https://py-readability-metrics.readthedocs.io/en/latest/

Drif, A., Hamida, Z. F., & Giordano, S. (2019). Fake News Detection Methdo Based on Text-Features. *MMM 2019 : The Ninth International Conference on Advances in Information Mining and Management* (pp. 1 - 6). Nice: IARIA XPS Press.

Dyson, L., & Golab, A. (2017). *Exploring the Application of NLP Methods to Machine Identification of Misleading News Sources.* Github. Retrieved June 20, 2021, from https://github.com/aldengolab/fake-news-detection/raw/master/fakenews_finalpaper.pdf

Earl, B. (2010). *The Practise of Social Research.* Cengage Publishing.

Elhadad, M. K., Fun Li, K., & Gebali, F. (2020). Detecting Misleading Information on COVID-19. *IEEE Access, 8*, 165201-165215. doi:10.1109/ACCESS.2020.3022867

Fake News Challenge. (2016). *Fake News Challenge*. Retrieved from Fake News Challenge: http://www.fakenewschallenge.org/

Fernandez, P. (2017). The technology behind fake news. *Library Hi Tech News, 34*(7), 1 - 5. doi:doi.org/10.1108/LHTN-07-2017-0054

Ferreira, W., & Vlachos, A. (2016). Emergent: A novel data-set for stance classification. *016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 1163 - 1168).

Geron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & TensorFlow.* O' Reilly Media Inc. Retrieved July 02, 2021

Goldani, M. H., Safabakhsh, R., & Momtazi, S. (2021). Convolutional neural network with margin loss for fake news detection. *Information Processing and Management 58*, 1 - 12.

Goldstuck, A., & Patricios, O. (2017, September 21). *SA Social Media Landscape 2018 - Executive Summary.* Retrieved March 23, 2018, from WorldWideWorx: http://website.ornico.co.za/2017/09/sa-social-media-2018/

Google. (2018, March 20). *Our Partnerships*. Retrieved July 02, 2021, from Google News Initatives: https://newsinitiative.withgoogle.com/partnerships/

Gottfried, J., & Shearer, E. (2017, September 7). *News Use Across Social Media Platforms 2017*. Retrieved from Journalism: https://www.journalism.org/2017/09/07/news-use-across-social-media-platforms-2017/

Granik, M., & Mesyura, V. (2017). Fake News Detection Using Naive Bayes Classifier. *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)* (pp. 900 - 903). Kyiv: IEEE. doi:10.1109/UKRCON.2017.8100379

Granskogen, T. (2018). *Automatic Detection of Fake News in Social Media using Contexual Information.* Norweigian University of Science and Technology.

References

Gravanis, G., Vakali, A., Diamantaras, K., & Karadais, P. (2019). Behind the cues: A benchmarking study for fake news detection. *Expert Systems With Applications*, 201 - 213.

Gravanis, G., Vakali, A., Diamantaras, K., & Karadais, P. (2019). Behind the cues: A benchmarking study for fake news detection. *Expert Systems With Applications, 128*, 201 - 213. doi:10.1016/j.eswa.2019.03.036

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques.* Elsevier. doi:https://doi.org/10.1016/C2009-0-61819-5

Hanselowski, A., PVS, A., Schiller, B., & Caspelherr, F. (2017). *Description of the System Developed by Team.* GitHub. Retrieved October 11, 2021, from https://github.com/hanselowski/athene_system/blob/master/system_description_athene.pdf

Hassan, N., Arslan, F., Li, C., & Tremayne, M. (2017). Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17* (pp. 1 - 10). Association for Computing Machinery, New York, NY, United States. doi:10.1145/3097983.3098131

Hassan, N., Arslan, F., Li, C., & Tremayne, M. (2017). Toward Automated Fact-Checking: Detecting Check-Worthy Factual Claims by ClaimBuster. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, 1803 - 1812.

Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jmenez, D., Gawsane, S., . . . Tremayne, M. (2017). ClaimBuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, (pp. 1945 - 1948). Retrieved from https://dl.acm.org/doi/10.14778/3137765.3137815

Horne, B. D., & Adali, S. (2017). This Just In: Fake News Packs a Lot in Title, Uses Simpleer Reptitive Content in Text Body, More Similiar to Satire than Real Newsq. *Arxiv*, 1 - 9.

References

Horne, B. D., & Adali, S. (2017). This Just In: Fake News Packs a Lot in Title, Uses Simpler, Repetitive Content in Text Body, More Similar to Satire than Real News. *arXiV* (pp. 1 - 9). Cornell University. doi:arXiv:1703.09398

Hossain, T., IV, R. L., Ugarte, A., Matsubara, Y., Young, S., & Singh, S. (2020). COVIDLies: Detecting COVID-19 Misinformation on Social Media. *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020* (pp. 1- 11). Association for Computational Linguistics. doi:10.18653/v1/2020.nlpcovid19-2.11

Hugging Face. (2020). *T5 - Transformers 4.3.0 documentation*. Retrieved February 27, 2021, from Huggingface: https://huggingface.co/transformers/model_doc/t5.html

Hyman, J. (2018). Addressing fake news: Open Standards & easy identification. *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2017*. doi:https://doi.org/10.1109/UEMCON.2017.8248986

Jaidka, K., Khoo, C. S., & Na, J.-C. (2013). Literature review writing: how information is selected and transformed. *Aslib Proceedings, 65*(3), 303 - 325. doi:10.1108/00012531311330665

Jiang, T., Ping Li, J., Ul Haq, A., Saboor, A., & Ali, A. (2021). A Novel Stacking Approach for Accurate Detection of Fake News. *IEEE Access, 9*, 22627 - 22639.

Kaggle. (2018). *Fake News | Kaggle*. Retrieved May 01, 2021, from Kaggle: https://www.kaggle.com/c/fake-news/data

Kaliyah, R. K., Goswami, A., Narang, P., & Sinha, S. (2020). FNDNet – A deep convolutional neural network for fake news detection. *Cognitive Systems Research, 61*, 32 - 44. doi:10.1016/j.cogsys.2019.12.005

Kaliyar, R. K., Goswami, A., & Narang, P. (2020). DeepFakE: improving fake news detection using tensor decomposition-based deep neural network. *The Journal of Supercomputing*, 1 - 23. doi:https://doi.org/10.1007/s11227-020-03294-y

Kawa, L., & Goko, C. (2018, January 9). *How fake news and Elon Musk sent the rand haywire*. Retrieved July 13, 2018, from Moneyweb:

https://www.moneyweb.co.za/news/markets/how-fake-news-and-elon-musk-sent-the-rand-haywire/

Keras. (2020, May 09). *Layer Activation Functions*. Retrieved August 09, 2021, from Keras: https://keras.io/api/layers/activations/

Keras. (2020, September 20). *Probabalistic Losses*. Retrieved August 09, 2021, from Keras: https://keras.io/api/losses/probabilistic_losses/#binarycrossentropy-class

Keras. (2021, March 05). *GRU Layer*. Retrieved March 22, 2021, from Keras: https://keras.io/api/layers/recurrent_layers/gru/

Keras-Tuner. (2019, November 16). *Keras Tuner*. Retrieved May 08, 2021, from KerasTuner: https://keras-team.github.io/keras-tuner/

Khodabandehlou, S., & Rahman, M. Z. (2017). Comparison of supervised machine learning techniques for customer churn prediction based on analysis of customer behavior. *Journal of Systems and Information Technology*, 65 - 93.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *arXiv*, 1 - 6.

Kogan, S., Moskowitz, T. J., & Niessner, M. (2019). *Fake News: Evidence from Financial Markets.* SSRN. doi:https://dx.doi.org/10.2139/ssrn.3237763

Kshetri, N., & Voas, J. (2017). The Economics of Fake News. *IT Professional, 19*(6), 8 - 12.

Kula, S., Choras, M., Kozik, R., Ksieniewicz, P., & Wozniak, M. (2020). Sentiment Analysis for Fake News Detection by Means of Neural Networks. *International Conference on Computational Science 2020* (pp. 653 - 666). Springer. doi:https://doi.org/10.1007/978-3-030-50423-6_49

Kula, S., Kozik, R., & Choras, M. (2021). Implementation of the BERT-derived architectures to tackle. *Neural Computing and Applications*, 1 - 14. Retrieved from https://doi.org/10.1007/s00521-021-06276-0

Kula, S., Kozik, R., Choras, M., & Wozniak, M. (2021). Transformer Based Models in Fake News Detection. In M. Paszynski, D. Kranzlmüller, V. V.

Krzhizhanovskaya, J. J. Dongarra, & P. M. Sloot, *Computational Science – ICCS 2021* (pp. 28 - 38). Krakow, Poland: Springer.

Lazer, D., Baum, M., Benkler, Y., Berinsky, A., Greenhill, K., Menczer, F., . . . Zittrain, J. (2018). The science of fake news. *Science*, 1094 - 1096.

le Roux, J. (2018, November 15). *EXPOSED: The Unisa employee who manufactures fake news to divide SA*. Retrieved April 14, 2019, from News24: https://www.news24.com/SouthAfrica/News/how-fake-news-grabbed-hold-of-the-dros-rape-20181115

Li, Q., Li, P., & Lo, E. Y.-M. (2020). Improving convolutional neural network for text classification by recursive data pruning. *Neurocomputing*, 143 - 152.

Mahudeswaran, D., & Shu, K. (2019). *FakeNewsNet | Kaggle*. Retrieved May 01, 2021, from Kaggle: https://www.kaggle.com/mdepak/fakenewsnet

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Support vector machines and machine learning on documents. In C. D. Manning, P. Raghavan, & H. Schütze, *Introduction to Information Retrieval* (pp. 320 - 348). Cambridge University Press. doi:https://doi.org/10.1017/CBO9780511809071

Masood, R., & Aker, A. (2018). The Fake News Challenge: Stance Detection using Traditional Machine Learning Approaches. *IC3K 2018 - Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management* (pp. 128 - 135). Springer. doi:10.5220/0006898801280135

Matsa, K. E., & Shearer, E. (2018, September 10). *News Use Across Social Media Platforms 2018*. Retrieved March 24, 2018, from Journalism: https://www.journalism.org/2018/09/10/news-use-across-social-media-platforms-2018/

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations ofWords and Phrases and their Compositionality. *arXiv*, 1 - 9.

Misback, E., & Pfeifer, C. (2017, Feburary 1). *FakeNewsChallenge/fnc-1*. Retrieved from GitHub: https://github.com/FakeNewsChallenge/fnc-1

References

Mohaddad. (2020, August 29). *GitHub - mohaddad/COVID-FAKES: Bilingual (Arabic/English) COVID-19 Twitter dataset for misleading information detection*. Retrieved May 01, 2021, from Github: https://github.com/mohaddad/COVID-FAKES

Mosseri, A. (2017, April 06). *Working to Stop Misinformation and False News*. Retrieved July 09, 2021, from Facebook: https://about.fb.com/news/2017/04/working-to-stop-misinformation-and-false-news/

Mueller, A. (2020, November 11). *wordcloud PyPI*. Retrieved May 15, 2021, from pypi: https://pypi.org/project/wordcloud/

MyBroadband. (2017, March 27). *List of Known Fake News Sites in South Africa (and beyond)*. Retrieved April 07, 2019, from MyBroadband: https://mybroadband.co.za/forum/threads/list-of-known-fake-news-sites-in-south-africa-and-beyond.879854/

Nasir, J. A., Khan, O. S., & Varlamis, I. (2021, January 5). Fake news detection: A hybrid CNN-RNN based deep learning approach. *International Journal of Information Management Data Insights, 1*(1), 1 - 13. doi:https://doi.org/10.1016/j.jjimei.2020.100007

NLTK. (2021, April 20). *NLTK 3.6.2 documentation*. Retrieved May 12, 2020, from NLTK: https://www.nltk.org/

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., & Invernizzi, L. (2019). *Keras-Tuner*. Retrieved May 01, 2021, from Keras-Tuner: https://github.com/keras-team/keras-tuner

OpenSources. (2018, January 10). *OpenSources*. Retrieved September 26, 2018, from OpenSources: http://www.opensources.co/

Pandas. (2012, February 13). *Pandas - Python Data Analysis Library*. Retrieved May 12, 2021, from Pandas: https://pandas.pydata.org/

Pathak, A., & Srihari, R. (2019). BREAKING! Presenting Fake News Corpus For Automated Fact Checking. *57th Annual Meeting ofthe Association for Computational Linguistics: Student Research Workshop* (pp. 357 - 362).

Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-2050

Patwa, P., Sharma, S., PYKL, S., Guptha, V., Kumari, G., Akhtar, M. s., . . . Chakraborty, T. (2020). Fighting an infodemic: COVID-19 fake news dataset. *arXiv*, 1- 5. doi:10.1007/978-3-030-73696-5_3

Phung, V., & Rhee, E. (2019). A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*, 1 - 16. Retrieved July 11, 2021, from https://doi.org/10.3390/app9214500

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. (pp. 1 - 12). OpenAI.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., . . . Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified. *Journal of Machine Learning Research 21*, 1 - 67.

Rashkin, H., Choi, E., Yea Jang, J., Volkova, S., & Choi, Y. (2017). Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2931 - 2937. doi:10.18653/v1/D17-1317

Robb, M. B. (2017). *News and America's kids: How young people perceive and are impacted by the news.* San Francisco: Common Sense Media.

Roberts, A., & Raffel, C. (2020, February 24). *Exploring Transfer Learning with T5: the Text-To-Text Transfer Transformer*. Retrieved October 10, 2021, from Google AI Blog: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

Rodny-Gumede, Y. (2018). Fake It till You Make It: The Role, Imapct and Consequences of Fake News. In B. Mutsvairo, & B. Karam, *Perspectives on Political Communication in Africa* (pp. 203 - 211). Cham: Springer Nature.

Rodrigues, U. M., & Xu, J. (2020). Regulation of COVID-19 fake news infodemic in China and India. *Media International Australia, 177*(1), 125-131. doi:10.1177/1329878X20948202

References

Rubin, V. L., Brogly, C., Conroy, N., Chen, Y., Cornwell, S. E., & Asubiaro, T. V. (2019). A News Verification Browser for the Detection of Clickbait, Satire, and Falsified News. *The Journal of Open Source Software*, 1 - 4. Retrieved from https://doi.org/10.21105/joss.01208

Rubin, V. L., Chen, Y., & Conroy, N. j. (2015). Deception Detection for Fake News: Three Types of Fakes. *Proceedings of the Association for Information Science and Technology*, 1 - 4.

Ryan, G. (2018). Introduction to positivism, interpretivism and critical theory. *Nurse Researcher 25(4), 25*(4), 14 - 20. doi:10.7748/nr.2018.e1466

SAGE. (2020, February 12). *Access Coronavirus (COVID-19) Articles from SAGE Publishing | SAGE Publications Ltd.* Retrieved May 02, 2021, from SAGE: https://journals.sagepub.com/pb-assets/PDF/SAGE-Publishing_Coronavirus-Related-Articles-1598635698300.pdf

SAGE. (2020, February 12). *Acess Coronavirus (COVID-19) Articles from SAGE Publishing | SaGE Publications Ltd*. Retrieved May 02, 2021, from SAGE Publishing: https://uk.sagepub.com/en-gb/eur/press/access-coronavirus-covid-19-articles-from-sage-publishing

Sahoo, S. R., & Gupta, B. B. (2021). Multiple features based approach for automatic fake news detection on social networks using deep learning. *Applied Soft Computing Journal*, 1 - 16. doi:https://doi.org/10.1016/j.asoc.2020.106983

Saquete, E., Tomas, D., Moreda, P., Martinez-Barco, P., & Palomar, M. (2020). Fighting post-truth using natual language processing: A review and open challenges. *Expert Systems With Applications*, 1 - 27.

Sarkar, A. K., & Sahu, T. N. (2018). *Investment Behaviour: Towards an Individual-Centred Financial Policy in Developing Economies.* Emerald Publishing Limited. doi:https://www.emeraldinsight.com/doi/abs/10.1108/978-1-78756-279-020181004

Scerri, M., & Grech, V. (2020). Countering fake news in the COVID-19 era: The public's opinion on the role of an honest and reliable website. *Early Human Development*, 1 - 3. doi:https://doi.org/10.1016/j.earlhumdev.2020.105257

SciKit Learn. (2012, April 06). *Tuning the hyper-parameters of an estimator — scikit-learn 0.23.1 documentation*. Retrieved July 4, 2020, from SciKit Learn: https://scikit-learn.org/stable/modules/grid_search.html

SciKit-Learn. (2012, January 13). *1.10. Decision Trees - scikit learn 0.24.2 documentation*. Retrieved July 02, 2021, from Scikit Learn: https://scikit-learn.org/stable/modules/tree.html#tree

SciKit-Learn. (2012, November 15). *1.6. Nearest Neighbors - scikit learn 0.24.2 documentation*. Retrieved July 02, 2021, from SciKit Learn: https://scikit-learn.org/stable/modules/neighbors.html

SciKit-Learn. (2012, June 16). *sklearn.feature_extraction.text.TfidfTransformer - scikit learn 1.0 documentation*. Retrieved from scikit learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer

SciKit-Learn. (2020, September 18). *sklearn.ensemble.RandomForestClassifier - scikit learn 0.24.2 documentation*. Retrieved July 02, 2021, from SciKit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ensemble.RandomForestClassifier

SciKit-Learn. (2021, August 04). *sklearn.metrics.f1_score - scikit-learn 0.24.2 documentation*. Retrieved August 04, 2021, from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

SciKit-Learn. (2021, August 03). *sklearn.metrics.precision_score - scikit-learn 0.24.2 documentation*. Retrieved August 04, 2021, from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

SciKit-Learn. (2021, August 2021). *sklearn.metrics.recall_score - scikit-learn 0.24.2 documentation*. Retrieved August 04, 2021, from scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

Sharma, S., Sharma, S., & Athaiya, A. (2020). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology, 4*(12), 310 - 316.

Shen YS, L. (2018, May 09). *lexicalrichness PyPI*. Retrieved May 12, 2021, from pypi: https://pypi.org/project/lexicalrichness/

Shu, K., Mahudeswaran, D., & Liu, H. (2018). FakeNewsTracker: a tool for fake news collection, detection and visualization. *Computational and Mathematical Organization Theory*, 1 - 13. doi:10.1007/s10588-018-09280-3

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2018). FakeNewsNet: A Data Repository with News Content, Social Context and and Dynamic Information for Studying Fake News on Social Media. *arXiv*, 1 - 12.

SignalMedia. (2015, September 01). *NewsIR'16 - Signal Media News Dataset*. Retrieved July 20, 2019, from Signal Media: https://research.signal-ai.com/newsir16/signal-dataset.html

Silva, R. M., Santos, R. L., Almeida, T. A., & Pardo, T. A. (2020). Towards automatically filtering fake news in Portuguese. *Expert Systems With Applications*, 1 - 14. doi:https://doi.org/10.1016/j.eswa.2020.113199

Silverman, C. (2015, February 10). *Lies, Damn Lies, and Viral Content*. Retrieved from Columbia Journalis Review: https://www.cjr.org/tow_center_reports/craig_silverman_lies_damn_lies_viral_content.php

Silverman, C. (2016, November 16). *This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook*. Retrieved June 22, 2019, from BuzzFeed News: https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook

Singhal, S., Chakraborty, T., Kumaraguru, P., Satoh, S., & Shah, R. R. (2019). SpotFake: A Multi-modal Framework for Fake News Detection. *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)* (pp. 40 - 47). Singapore: IEEE. doi:https://dx.doi.org/10.1109/BigMM.2019.00-44

Slovikovskaya, V., & Attardi, G. (2020). Transfer Learning from Transformers to Fake News Challenge Stance Detection (FNC-1) Task. *Proceedings of the 12th Conference on Language Resources and Evaluation*, (pp. 1211–1218).

South African Government. (2020, September 24). *Fake News - Coronavirus COVID-19 | South African Government*. Retrieved from South African Government: https://www.gov.za/covid-19/resources/fake-news-coronavirus-covid-19

Sreekumar, D., & Chitturi, B. (2019). Deep neural approach to Fake-News identificatio. *International Conference on Computational Intelligence and Data Science (ICCIDS 2019)* (pp. 2236-2243). Elsevier. doi:10.1016/j.procs.2020.03.276

Stanescu, A., Mata-Toledo, R. A., & Gupta, P. (2018). Machine Learning. *AccessScience. McGraw-Hill Education*. doi:https://doi.org/10.1036/1097-8542.395250

Statistica. (2018). *Number of monthly active Twitter users worldwide from 1st quarter 2010 to 2nd quarter 2018 (in millions)*. Retrieved September 25, 2018, from Statistica: https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

Stickel, M., & Tardo, F. (2020, April 7). *IEEE - IEEE Provides Free Access to COVID-19 Relevant Research Articles and Standards in the IEEE Xplore Digital Library*. Retrieved May 02, 2021, from IEEE Xplore: https://www.ieee.org/about/news/2020/ieee-provides-free-access.html

Stockling, G., Barthel, M., & Grieco, E. (2018, January 29). *Sources Shared on Twitter: A Case Study on Immigration | Pew Research Center*. Retrieved September 26, 2018, from Pew Research Center: http://www.journalism.org/2018/01/29/sources-shared-on-twitter-a-case-study-on-immigration/

Subramanian, S., & Martin, G. (2017, February 15). *The Macednonian Teens Who Mastered Fake News*. Retrieved September 26, 2018, from Wired: https://www.wired.com/2017/02/veles-macedonia-fake-news/

Sydell, L. (2016, November 23). *We Tracked Down A Fake-News Creator In The Suburbs. Here's What We Learned*. Retrieved June 22, 2019, from National

Public Radio (NPR): https://www.npr.org/sections/alltechconsidered/2016/11/23/503146770/npr-finds-the-head-of-a-covert-fake-news-operation-in-the-suburbs

Szpakowski, M. (2018, February 02). *several27/FakeNewsCorpus: A dataset of millions of news articles scraped from a curated list of data sources.* Retrieved July 20, 2019, from GitHub: https://github.com/several27/FakeNewsCorpus

Tagliabue, F., Galassi, L., & Mariani, P. (2020). The "Pandemic" of Disinformation in COVID-19. *SN Comprehensive Clinical Medicine*, 1287–1289. doi:10.1007/s42399-020-00439-1

Tanvir, A. A., Mahir, E. M., Akhter, S., & Huq, M. R. (2019). Detecting Fake News using Machine Learning and Deep Learning Algorithms. *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, (pp. 1 - 5).

Tchakounté, F., & Hayata, F. (2016). Chapter 6 - Supervised Learning Based Detection of Malware on Android. In M. H. Au, & K.-K. R. Choo, *Mobile Security and Privacy - Advances, Challenges and Future Research Directions* (pp. 101 - 154). Retrieved from https://doi.org/10.1016/C2015-0-00278-7

Van Eemeren, F. H., & Grootendorst, R. (2004). *A Systematic Theory of Argumentation.* Cambridge: Cambridge University Press. Retrieved September 26, 2018, from http://catdir.loc.gov/catdir/samples/cam041/2003046181.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 5999-6009. Retrieved from https://arxiv.org/abs/1706.03762

Vierra, S., Pinaya, W. H., & Mechelli, A. (2019). *Machine Learning: Methods and Applications to Brain Disorders.* (A. Mechelli, & S. Viera, Eds.) London, United Kingdom: Academic Press.

Vishwakarma, D. K., Varshney, D., & Yadav, A. (2019). Detection and veractiy analysis of fake news via scrapping and authenticating the web search. *Cognitive Systems Research*, 217 - 229.

Wasserman, H. (2017). Fake news from Africa: Panics, politics and paradigms. *Journalism: Theory, Practice & Criticism*, 1 - 14. doi:10.1177/1464884917746861

Western Cape Government. (2020, June 16). *Fake News | COVID 19 response*. Retrieved January 05, 2021, from Western Cape Government: https://coronavirus.westerncape.gov.za/fake-news

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. M. (2019). HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv*, 1 - 8. doi:10.18653/v1/2020.emnlp-demos.6

Wu, X., Cheng, S., & Chai, Z. (2017). Fake News Stance Detection. 1 - 6.

Yang, Y., Zheng, L., Zhang, J., Cui, Q., Zhang, X., Li, Z., & Yu, P. S. (2018). TI-CNN: Convolutional Neural Networks for Fake News Detection. *arXiv*, 1 - 13.

YS, L. S. (2018, May 09). *lexicalrichness - PyPI*. Retrieved May 01, 2021, from PyPI: https://pypi.org/project/lexicalrichness/

Zhang, C., Gupta, A., Kauten, C., Deokar, A. V., & Qin, X. (2019). Detecting gake news for reducing misinformation risks using analytics approaches. *European Journal of Operational Research*, 1036 - 1052.

Zhang, J., Dong, B., & Yu, P. S. (2016). FAKEDETECTOR: Effective Fake News Detection with Deep Diffusive Neural Network. *arXiv*, (pp. 1 - 13). Retrieved from https://arxiv.org/abs/1805.08751

Zhang, X., & Ghorbani, A. A. (2019). An overview of online fake news: Characterization, detection and discussion. *Information Processing and Management*, 2 - 10. doi:https://doi.org/10.1016/j.ipm.2019.03.004

Zhang, Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of Translational Medicine*, 1 - 7. doi:10.21037/atm.2016.03.37

Zhu, S., & Chollet, F. (2019, April 14). *Working with RNNs*. Retrieved January 23, 2020, from Keras: https://keras.io/guides/working_with_rnns/#using-cudnn-kernels-when-available

References

Zimdar, M. (2016). *False, Misleading, Clickbait-y, and/or Satirical “News” Sources.* Retrieved September 25, 2018, from https://docs.google.com/document/d/10eA5-mCZLSS4MQY5QGb5ewC3VAL6pLkT53V_81ZyitM/preview

Appendix A - Academic Publications

This section lists all publications related to this study

Fake News Detection Using Content-Based Features and Machine Learning

O. Ngada and B. Haskins, "Fake News Detection Using Content-Based Features and Machine Learning," 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), 2020, pp. 1-6, DOI: 10.1109/CSDE50874.2020.9411638.

Abstract: The problem of fake news is a complex problem and is accompanied by social and economic ramifications. Targeted individuals and entities may lose trustworthiness, credibility and ultimately, suffer from reputation damages to their brand. Economically, an individual or brand may see fluctuations in revenue streams. In addition, the complex nature of the human language makes the problem of fake news a complex problem to solve for currently available computational remedies. The fight against the spread of fake news is a multi-disciplinary effort that will require research, collaboration and rapid development of tools and paradigms aimed at understanding and combating false information dissemination. This study explores fake news detection techniques using machine learning technology. Using a feature set that captures article structure, readability, and the similarity between the title and body, we show such features can deliver promising results. In the experiment, we select 6 machine learning algorithms, namely, AdaBoost as AB, Decision Tree as DT, K-Nearest Neighbour as KNN, Random Forest as RF, Support Vector Machine as SVM and XGBoost as XGB. To quantify a classifier's performance, we use the confusion matrix model and other performance metrics. Given the structure of the experiment, we show the Support Vector Machine classifier provided the best overall result.

Appendix B - Hybrid Convolutional Neural Network Experimental Data

Hybrid CNN Training Results

Table B.1: Training and validation results obtained for the hybrid CNN model over 20 epochs

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
1	95,00%	0,1119	99,87%	0,0078
2	99,77%	0,093	99,81%	0,0086
3	99,83%	0,0065	99,88%	0,0071
4	99,85%	0,0047	99,86%	0,0091
5	99,87%	0,0049	99,86%	0,0092
6	99,82%	0,0066	99,88%	0,0081
7	99,96%	0,0019	99,86%	0,0140
8	99,93%	0,0020	99,91%	0,0091
9	99,87%	0,0046	99,91%	0,0085
10	99,90%	0,0041	99,92%	0,0084
11	99,94%	0,0028	99,86%	0,0123
12	99,93%	0,0022	99,90%	0,0132
13	99,91%	0,0041	99,89%	0,0141
14	99,95%	0,0019	99,89%	0,0130
15	99,95%	0,0016	99,88%	0,0131
16	99,96%	0,0015	99,86%	0,0164
17	99,95%	0,0019	99,87%	0,0181
18	99,96%	0,0018	99,88%	0,0192
19	99,91%	0,0036	99,88%	0,0241
20	99,93%	0,0030	99,89%	0,0242

Table B.2: Confusion Matrix results obtained at each epoch of training the neural network model, using the training portion of data (80% of data)

Epoch	TP	TN	FP	FN	AUC	Precision	Recall
1	17696	14805	2247	1170	97,22%	88,73%	93,80%
2	18465	16338	714	401	99,55%	96,28%	97,87%
3	18538	16528	524	328	99,73%	97,25%	98,26%
4	18560	16564	488	306	99,77%	97,44%	98,38%
5	18580	16631	421	286	99,79%	97,78%	98,48%
6	18604	16690	362	262	99,85%	98,09%	98,61%
7	18601	16668	384	265	99,84%	97,98%	98,60%
8	18630	16720	332	236	99,86%	98,25%	98,75%
9	18629	16732	320	237	99,86%	98,31%	98,74%
10	18648	16739	313	218	99,88%	98,35%	98,84%
11	18641	16752	300	225	99,87%	98,42%	98,81%
12	18652	16777	275	214	99,87%	98,55%	98,87%
13	18686	16776	276	180	99,90%	98,54%	99,05%
14	18672	16775	277	194	99,89%	98,54%	98,97%
15	18670	16793	259	196	99,90%	98,63%	98,96%
16	18685	16779	273	181	99,88%	98,56%	99,04%
17	18698	16797	255	168	99,91%	98,65%	99,11%
18	18694	16805	247	172	99,91%	98,70%	99,09%
19	18684	16787	265	182	99,91%	98,60%	99,04%
20	18686	16782	270	180	99,90%	98,58%	99,05%

Table B.3: Confusion Matrix results obtained at each epoch of training the neural network model, using the validation portion of data (20% of data)

Epoch	TP	TN	FP	FN	AUC	Precision	Recall
1	4594	4358	7	21	100,00%	99,85%	99,54%
2	4610	4357	8	5	100,00%	99,83%	99,89%
3	4608	4361	4	7	99,99%	99,91%	99,85%
4	4596	4365	0	19	100,00%	100,00%	99,59%
5	4611	4362	3	4	100,00%	99,93%	99,91%
6	4597	4365	0	18	100,00%	100,00%	99,61%
7	4611	4362	3	4	100,00%	99,93%	99,91%
8	4604	4363	2	11	100,00%	99,96%	99,76%
9	4612	4364	1	3	100,00%	99,98%	99,93%
10	4604	4365	0	11	100,00%	100,00%	99,76%
11	4607	4364	1	8	100,00%	99,98%	99,83%
12	4614	4362	3	1	100,00%	99,94%	99,98%
13	4611	4363	2	4	100,00%	99,96%	99,91%
14	4606	4364	1	9	100,00%	99,98%	99,80%
15	4606	4363	2	9	100,00%	99,96%	99,80%
16	4607	4365	0	8	100,00%	100,00%	99,83%
17	4610	4365	0	5	100,00%	100,00%	99,89%
18	4613	4363	2	2	100,00%	99,96%	99,96%
19	4605	4363	2	10	100,00%	99,96%	99,78%
20	4602	4363	2	13	100,00%	99,96%	99,72%

Appendix C - Convolutional Neural Network Epxerimental Data

CNN Training Results

Table C.1: Training and validation results obtained using the best configuration determined by the hyperparameter selection process

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
1	87,70%	0,2538	96,66%	0,0876
2	97,19%	0,0796	97,30%	0,0753
3	98,16%	0,0530	98,73%	0,0406
4	98,53%	0,0397	98,55%	0,0421
5	98,94%	0,0317	98,48%	0,0450
6	99,00%	0,0274	99,01%	0,0319
7	99,24%	0,0251	99,09%	0,0255
8	99,23%	0,0214	99,06%	0,0278
9	99,32%	0,0184	99,24%	0,0267
10	99,40%	0,0169	99,24%	0,0263
11	99,41%	0,0170	99,32%	0,0249
12	99,43%	0,0163	99,15%	0,0291
13	99,51%	0,0132	99,29%	0,0243
14	99,50%	0,0148	98,94%	0,0403
15	99,47%	0,0161	98,63%	0,0467
16	99,61%	0,0125	99,21%	0,0327
17	99,65%	0,0108	98,96%	0,0412
18	99,60%	0,0113	99,29%	0,0271
19	99,55%	0,0127	99,44%	0,0230
20	99,77%	0,0081	99,40%	0,0267

Table C.2: AUC, Precision, Recall and confusion matrix results obtained at each epoch using the training portion of the dataset

Epoch	AUC	Precision	Recall	TN	TP	FN	FP
1	95,78%	85,24%	92,50%	12344	15217	1233	2634
2	99,54%	96,74%	97,92%	14436	16108	342	542
3	99,76%	97,95%	98,55%	14638	16212	238	340
4	99,86%	98,47%	98,73%	14726	16241	209	252
5	99,91%	98,94%	99,04%	14803	16292	158	175
6	99,92%	99,00%	99,09%	14813	16301	149	165
7	99,91%	99,27%	99,28%	14858	16331	119	120
8	99,95%	99,19%	99,34%	14844	16341	109	134
9	99,95%	99,33%	99,37%	14868	16346	104	110
10	99,97%	99,40%	99,45%	14880	16360	90	98
11	99,97%	99,45%	99,43%	14887	16356	94	91
12	99,96%	99,45%	99,47%	14887	16362	88	91
13	99,97%	99,53%	99,53%	14900	16373	77	78
14	99,96%	99,49%	99,55%	14894	16376	74	84
15	99,96%	99,46%	99,53%	14889	16373	77	89
16	99,97%	99,63%	99,62%	14917	16387	63	61
17	99,97%	99,67%	99,65%	14924	16393	57	54
18	99,97%	99,64%	99,59%	14919	16383	67	59
19	99,97%	99,55%	99,60%	14904	16384	66	74
20	99,98%	99,78%	99,78%	14942	16413	37	36

Table C.3: AUC, Precision, Recall and confusion matrix metrics results obtained at each epoch, using the test portion of the dataset

Epoch	AUC	Precision	Recall	TN	TP	FN	FP
1	99,53%	94,62%	99,25%	6042	6978	53	397
2	99,72%	95,27%	99,77%	6091	7015	16	348
3	99,83%	98,69%	98,88%	6347	6952	79	92
4	99,82%	97,67%	99,60%	6272	7003	28	167
5	99,84%	97,34%	99,82%	6247	7018	13	192
6	99,89%	98,40%	99,73%	6325	7012	19	114
7	99,92%	98,88%	99,39%	6360	6988	43	79
8	99,90%	98,58%	99,63%	6338	7005	26	101
9	99,88%	99,08%	99,46%	6374	6993	38	65
10	99,86%	99,05%	99,50%	6372	6996	35	67
11	99,89%	99,04%	99,66%	6371	7007	24	68
12	99,88%	98,83%	99,56%	6356	7000	31	83
13	99,91%	99,36%	99,29%	6394	6981	50	45
14	99,76%	98,17%	99,83%	6308	7019	12	131
15	99,77%	97,59%	99,84%	6266	7020	11	173
16	99,80%	98,75%	99,76%	6350	7014	17	89
17	99,80%	98,20%	99,84%	6310	7020	11	129
18	99,89%	98,97%	99,67%	6366	7008	23	73
19	99,89%	99,35%	99,59%	6393	7002	29	46
20	99,86%	99,15%	99,70%	6379	7010	21	60

Appendix D - Recurrent Neural Network (GRU) Experimental Data

RNN (GRU) Training Results

Table D.1: GRU model training results over 20 epochs

GRU Model Results				
Total Parameters		15 540 249		
Trainable parameters		136 449		
Non-Trainable parameters		15 403 800		
Epoch	**Accuracy**	**Loss**	**Validation Accuracy**	**Validation Loss**
1	82,65%	0,3663	88,19%	0,2785
2	88,59%	0,2684	88,88%	0,2581
3	89,55%	0,2469	88,59%	0,2755
4	89,96%	0,2374	89,24%	0,2501
5	90,30%	0,2295	89,12%	0,2639
6	90,79%	0,2221	89,38%	0,2425
7	90,88%	0,2163	89,29%	0,2475
8	90,82%	0,2147	89,30%	0,2490
9	91,09%	0,2113	89,33%	0,2468
10	91,04%	0,2139	89,37%	0,2532
11	90,78%	0,2185	89,33%	0,2531
12	90,79%	0,2160	89,37%	0,2475
13	90,19%	0,2334	88,60%	0,2530
14	90,40%	0,2260	89,04%	0,2537
15	90,24%	0,2323	88,63%	0,2580
16	89,76%	0,2388	87,63%	0,2727
17	89,65%	0,2432	88,32%	0,2711
18	89,00%	0,2586	88,05%	0,2867
19	88,43%	0,2682	87,94%	0,2755
20	89,00%	0,2579	88,12%	0,2712

Table D.2: Results obtained at each epoch, using the training portion of the dataset (80% of data)

Epoch	TP	TN	FP	FN	AUC	Precision	Recall
1	15505	13764	3443	2701	91,46%	81,83%	85,16%
2	16481	14891	2316	1725	95,52%	87,68%	90,53%
3	16635	15077	2130	1571	96,21%	88,65%	91,37%
4	16737	15120	2087	1469	96,47%	88,91%	91,93%
5	16783	15194	2013	1423	96,72%	89,29%	92,18%
6	16868	15283	1924	1338	96,92%	89,76%	92,65%
7	16893	15289	1918	1313	97,08%	89,80%	92,79%
8	16898	15265	1942	1308	97,11%	89,69%	92,82%
9	16934	15323	1884	1272	97,18%	89,99%	93,01%
10	16905	15334	1873	1301	97,14%	90,03%	92,85%
11	16864	15284	1923	1342	97,02%	89,76%	92,63%
12	16820	15330	1877	1386	97,09%	89,96%	92,39%
13	16770	15168	2039	1436	96,64%	89,16%	92,11%
14	16768	15245	1962	1438	96,82%	89,52%	92,10%
15	16819	15138	2069	1387	96,63%	89,05%	92,38%
16	16683	15104	2103	1523	96,45%	88,81%	91,63%
17	16606	15142	2065	1600	96,33%	88,94%	91,21%
18	16551	14968	2239	1655	95,86%	88,08%	90,91%
19	16481	14836	2371	1725	95,53%	87,42%	90,53%
20	16591	14927	2280	1615	95,87%	87,92%	91,13%

Table D.3: Results obtained at each epoch, using the validation portion of the dataset (20% of data)

Epoch	TP	TN	FP	FN	AUC	Precision	Recall
1	4285	3523	686	360	95,36%	86,20%	92,25%
2	4222	3647	562	423	95,91%	88,25%	90,89%
3	4418	3426	783	227	96,13%	84,95%	95,11%
4	4202	3699	510	443	96,16%	89,18%	90,46%
5	4399	3492	717	246	96,23%	85,99%	94,70%
6	4298	3616	593	347	96,30%	87,88%	92,53%
7	4294	3612	597	351	96,17%	87,79%	92,44%
8	4273	3634	575	372	96,25%	88,14%	91,99%
9	4152	3757	452	493	96,37%	90,18%	89,39%
10	4313	3600	609	332	96,16%	87,63%	92,85%
11	4309	3600	609	336	96,14%	87,62%	92,77%
12	4286	3627	582	359	96,18%	88,04%	92,27%
13	4244	3601	608	401	95,94%	87,47%	91,37%
14	4243	3641	568	402	96,00%	88,19%	91,35%
15	4313	3534	675	332	95,78%	86,47%	92,85%
16	3954	3805	404	691	95,69%	90,73%	85,12%
17	4071	3749	460	574	95,65%	89,85%	87,64%
18	4236	3560	649	409	95,16%	86,71%	91,19%
19	4234	3552	657	411	95,33%	86,57%	91,15%
20	4242	3560	649	403	95,41%	86,73%	91,32%

Appendix E - Hybrid RNN (LSTM) Experimental Data

Hybrid RNN Training Data

Table E.1: Training Accuracy and Loss for the CNN-RNN hybrid model over 20 epochs

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
1	77,88%	0,4534	86,48%	0,3108
2	85,93%	0,32100	87,75%	0,2764
3	87,12%	0,2904	87,66%	0,2792
4	87,87%	0,2791	88,45%	0,2612
5	88,31%	0,2683	88,47%	0,2607
6	88,59%	0,2628	88,72%	0,2516
7	88,80%	0,2561	86,53%	0,2978
8	89,16%	0,2486	89,27%	0,2484
9	89,32%	0,2440	89,52%	0,2464
10	89,55%	0,2393	89,56%	0,2478
11	89,63%	0,2357	89,36%	0,2485
12	89,85%	0,2316	89,42%	0,2484
13	90,15%	0,2303	89,45%	0,2472
14	89,98%	0,2266	89,00%	0,2508
15	90,44%	0,2226	89,43%	0,2451
16	90,54%	0,2204	89,41%	0,2474
17	90,59%	0,2192	89,61%	0,2468
18	90,69%	0,2160	89,71%	0,2485
19	90,90%	0,2114	89,58%	0,2508
20	90,83%	0,2097	89,73%	0,2489

Table E.2: CNN-RNN performance evaluation during training at each epoch. 80% of articles were observed.

Epoch	TP	TN	FP	FN	AUC	Precision	Recall
1	15538	12040	5024	2811	86,43%	75,57%	84,68%
2	16309	14122	2942	2040	93,51%	84,72%	88,88%
3	16514	14338	2726	1835	94,67%	85,83%	90,00%
4	16534	14585	2479	1815	95,13%	86,96%	90,11%
5	16567	14707	2357	1782	95,50%	87,54%	90,29%
6	16680	14694	2370	1669	95,65%	87,56%	90,90%
7	16660	14788	2276	1689	95,90%	87,98%	90,80%
8	16716	14857	2207	1633	96,11%	88,34%	91,10%
9	16760	14871	2193	1589	96,25%	88,43%	91,34%
10	16782	14931	2133	1567	96,39%	88,72%	91,46%
11	16795	14947	2117	1554	96,49%	88,81%	91,53%
12	16822	14995	2069	1527	96,63%	89,05%	91,68%
13	16904	15020	2044	1445	96,67%	89,21%	92,12%
14	16830	15033	2031	1519	96,76%	89,23%	91,72%
15	16933	15095	1969	1416	96,86%	89,58%	92,28%
16	16966	15098	1966	1383	96,92%	89,62%	92,46%
17	17013	15068	1996	1336	96,94%	89,50%	92,72%
18	16993	15124	1940	1356	97,06%	89,75%	92,61%
19	17020	15169	1895	1329	97,17%	89,98%	92,76%
20	16984	15183	1881	1365	97,20%	90,03%	92,56%

Table E.3: CNN-RNN performance evaluation during testing at each epoch. 20% of articles were observed.

Epoch	TP	TN	FP	FN	AUC	Precision	Recall
1	4070	3587	765	432	94,34%	84,18%	90,40%
2	4097	3672	680	405	95,28%	85,77%	91,00%
3	4256	3505	847	246	95,65%	83,40%	94,54%
4	4007	3824	528	495	95,82%	88,36%	89,00%
5	4117	3716	636	385	95,93%	86,62%	91,45%
6	4166	3689	663	336	96,10%	86,27%	92,54%
7	3568	4093	259	934	96,01%	93,23%	79,25%
8	4089	3815	537	413	96,19%	88,39%	90,83%
9	4144	3782	570	358	96,26%	87,91%	92,05%
10	4157	3773	579	345	96,21%	87,77%	92,34%
11	4166	3746	606	336	96,26%	87,30%	92,54%
12	4212	3705	647	290	96,22%	86,68%	93,56%
13	4148	3772	580	354	96,34%	87,73%	92,14%
14	4000	3880	472	502	96,24%	89,45%	88,85%
15	4080	3838	514	422	96,30%	88,81%	90,63%
16	4196	3720	632	306	96,30%	86,91%	93,20%
17	4134	3800	552	368	96,29%	88,22%	91,83%
18	4103	3840	512	399	96,35%	88,91%	91,14%
19	4208	3723	629	294	96,34%	87,00%	93,47%
20	4186	3759	593	316	96,33%	87,59%	92,98%

Appendix F - Machine Learning Classifier Confusion Matrix Results

The table below shows confusion matrix related data collected through assessing the classification capabilities of the selected machine learning algoriths in chapter 5. A detailed explanation of each metric is discussed in section 3.10.

Table F.1: Counts on true positives, true negatives, false positives, false negatives for each classifier, using the test portion of the dataset

Classifier	TN	FP	FN	TP
Support Vector Machine	4248	29	27	4676
K-Nearest Neighbour	4115	151	639	4075
Random Forest	4268	41	49	4622
AdaBoost Classifier	4300	30	36	4614
XGBoost Classifier	4269	44	66	4601

Final Check thesis

ORIGINALITY REPORT

16%	10%	10%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Okuhle Ngada, Bertram Haskins. "Fake News Detection Using Content-Based Features and Machine Learning", 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), 2020 Publication	2%
2	link.springer.com Internet Source	1%
3	Deepak P, Tanmoy Chakraborty, Cheng Long, Santhosh Kumar G. "Data Science for Fake News", Springer Science and Business Media LLC, 2021 Publication	<1%
4	hdl.handle.net Internet Source	<1%
5	Submitted to University of West London Student Paper	<1%
6	dokumen.pub Internet Source	<1%
7	www.journaltocs.ac.uk	