Politechnika Wrocławska

Faculty of Management
Field of study: **Business Engineering**
Specialty: Business Intelligence

Master Thesis

Detection of fake news as an element of the information management system in the organization in the conditions of a war situation

Ahmad Megdadi

keywords:
natural language processing, machine learning,
neural network, Fake news, War times,
information management, Internet news portal

short summary:

Supervisor	dr hab. Eng. Zbigniew Michna		
	Title/ degree/ name and surname		
The final evaluation of the thesis			
Chairperson of the Diploma Examination Committee	....	....	....
	Title/ degree/ name and surname	*grade*	*Signature*

For the purposes of archival thesis qualified to: *

a) Category A (perpetual files)
b) Category BE 50 (subject to expertise after 50 years)

* Delete as appropriate

stamp of the faculty

Wrocław, 2023

Abstract

This master's thesis investigates the detection of fake news as an element of the information management system in organizations during war situations. The purpose of the thesis is to understand the unique challenges and characteristics that arise in detecting fake news during war and explore the role of information management and machine learning tools in internet news portals. The study examines the functionality of internet news portals, explores the use of machine learning tools in detecting fake news, and investigates variations in keyword usage for fake news detection in different war contexts. The thesis follows a structured approach, starting with an introduction to the significance of fake news in the digital age and war situations. The methodology section outlines the data cleaning process, including punctuation removal, tokenization, stop word removal, and lemmatization or stemming. The thesis concludes with proposed methods to enhance fake news detection during war on news portals, such as developing specialized datasets, domain-specific feature engineering, and exploring ensemble methods and deep learning models. Implementing these strategies can improve information management and mitigate the spread of fake news during war, promoting information accuracy and trustworthiness. The research contributes valuable insights and recommendations for detecting fake news in different contexts, acknowledging the evolving nature of the field and the need for further research.

Abstrakt (polish version)

Niniejsza praca magisterska bada wykrywanie fałszywych wiadomości jako element systemu zarządzania informacjami w organizacjach w sytuacjach wojennych. Celem pracy jest zrozumienie unikalnych wyzwań i cech, które pojawiają się w wykrywaniu fałszywych wiadomości podczas wojny oraz zbadanie roli zarządzania informacjami i narzędzi uczenia maszynowego w internetowych portalach informacyjnych. Badanie analizuje funkcjonalność internetowych portali informacyjnych, bada wykorzystanie narzędzi uczenia maszynowego w wykrywaniu fałszywych wiadomości i bada różnice w użyciu słów kluczowych do wykrywania fałszywych wiadomości w różnych kontekstach wojennych. Teza opiera się na ustrukturyzowanym podejściu, zaczynając od zdefiniowania znaczenia fałszywych wiadomości w erze cyfrowej i sytuacjach wojennych. Rozdział o metodologii przedstawia proces czyszczenia danych, w tym usuwanie interpunkcji, tokenizację, zatrzymania usuwanie słów oraz lematyzację lub stemming. Praca kończy się propozycjami metod poprawy wykrywania fałszywych wiadomości podczas wojny na portalach informacyjnych, takimi jak opracowywanie specjalistycznych zbiorów danych, inżynieria cech specyficznych dla domeny oraz badanie metod zespołowych i modeli głębokiego uczenia. Wdrożenie tych strategii może poprawić zarządzanie informacją i złagodzić rozprzestrzenianie się fałszywych wiadomości podczas wojny, promując dokładność i wiarygodność informacji. Badania wnoszą cenne spostrzeżenia i zalecenia dotyczące wykrywania fałszywych wiadomości w różnych kontekstach, uznając ewoluujący charakter tej dziedziny i potrzebę dalszych badań.

Table of Contents

1. Introduction

In the present digital age, the phenomenon of fake news has taken centre stage in the realm of journalism, becoming a critical issue, especially in the context of war. The dissemination of propaganda and misinformation during periods of military conflict poses significant challenges for information management in organizations, particularly those operating in the news industry. Understanding how organizations function, gather information, and detect fake news during war situations is crucial for effective information management and safeguarding the integrity of news.

One such organization that plays a pivotal role in news dissemination is an internet news portal. These portals function as platforms that collect, curate, and deliver news content to the audience. In the face of war, internet news portals face unique challenges in gathering accurate information and combatting the spread of fake news. The advent of machine learning tools has provided promising avenues for detecting and mitigating fake news, but their effectiveness and applicability in war contexts require further exploration.

This research seeks to address these gaps by investigating the role of information management and the application of machine learning tools in an internet news portal during war situations. By examining the functionality and information gathering processes of the portal, we aim to understand the unique characteristics and challenges that arise during war. Additionally, we will explore the use of machine learning tools in detecting fake news and the differences in keyword usage between normal times and war times. Specifically, we will investigate if there are variations in the keywords used to detect fake news in different war contexts, such as the Iraq, Syrian and Ukrainian unfortunate wars.

1.1 News Agencies

A news agency (also known as a wire service or news service) is an organization that gathers news reports and sells them to subscribing news organizations, such as newspapers, magazines, and radio and television broadcasters. These agencies exist to provide news to publications or broadcasters, not directly to the public, see [19].

News agencies were created to provide newspapers with a constant stream of news from around the world, and they remain a primary source of news and photographs for many newspapers and magazines. With the growth of mass media in the 20th and 21st centuries, news agencies have also grown to serve broadcast and digital news organizations.

News agencies prepare articles that are easily used by other news organizations with little or no modification and are designed to be used immediately. They provide these articles in a steady stream to news organizations.

Prominent news agencies include the Associated Press (AP), Reuters, and Agence France-Presse (AFP). These organizations operate on a global scale, with bureaus around the world. There are also national news agencies, such as the Press Trust of India (PTI) and the Australian Associated Press (AAP), which focus primarily on news within their respective countries.

News agencies often use correspondents, journalists who live in a foreign city and report on the events happening there. These correspondents report on politics, economics, culture, and other important topics, providing a global perspective on these subjects.

1.2 Internet News Portals

An internet news portal is a website that gathers news from various sources and presents it in one place for readers see[19]. These portals often provide a mix of original content, aggregated

content from other news sources, and user-generated content. News portals are structured to deliver a variety of news items to their readers. They often source their information from different channels, including direct reporting, expert opinions, and news agencies like the Associated Press (AP) or the British Broadcasting Corporation (BBC).
In addition to news articles, many internet news portals offer other features such as opinion pieces, blogs, video content, and interactive features. Some portals also allow users to customize their news feed, selecting the topics they are most interested in
Internet news portals have become a significant source of news for many people, particularly with the rise of digital media and the decline of traditional newspapers. They have the advantage of being able to provide up-to-the-minute news, as well as a wide variety of viewpoints and sources.

Prominent examples of internet news portals include Yahoo News, Google News, and MSN News. These sites aggregate news from various sources and present it in a user-friendly format. In peace times, news portals receive information from traditional news sources, social media, and other online sources. However, research that some portals may not always verify the information they publish, leading to the propagation of fake news. The desire to attract attention and pursue their own interests can lead to the publication of false information.
During war times, the process of news gathering became more complex. Reporters on the ground face several challenges, including personal security risks, basic physical needs, and operating within a complex information environment. Some organizations, like the Peace Journalism Lab (PJL), see[10], offer training programs for journalists to cope with these challenges. These programs cover several topics, including reporting on war and crises, field safety, data security, health, epidemics, stress management, and more. Many internet news portals aggregate news from various sources, selecting the most important stories from around the world and presenting them in an easily digestible format. Some of these portals use algorithms to select news stories based on relevance, timeliness, and popularity, while others employ human editors to curate the news.

It is worth mentioning that not all media outlets are the same, and some are more reliable than others. For example, the Associated Press (AP) is a non-profit organization that is not government-funded or corporate-sponsored. It often reports stories first, which other outlets then pick up. The British Broadcasting Corporation (BBC), funded by the British government, is the largest broadcaster in the world and has a long-standing reputation for accurate, unbiased reporting.

Covering news, especially in conflict zones, is a complex task that requires careful training. Aristotle University Thessaloniki, through its Peace Journalism Lab, offers training on field safety in war zones for journalists. The training covers several issues including the evolving nature of war reporting, field safety, data security, health, epidemics, and stress management. It seeks to provide journalists with the knowledge to cope with the challenges they might face in these contexts.

1.3 Fact Checking

Fact-checking is the process of verifying the factual accuracy of information see [8]. This is especially important in the context of news and journalism, where the spread of misinformation can have grave consequences. Fact-checking involves researching the claims made in a piece of content to determine if they are accurate. This can involve a variety of methods, including reviewing primary sources, speaking to experts, and using specialized fact-checking tools. Fact-checkers often work for news organizations or specialized fact-checking outlets. They review articles before they are published to ensure that all information is accurate and that any claims are supported by evidence.

In the context of internet news portals, fact-checking is crucial. These portals often aggregate news from a variety of sources, which can include both reputable news outlets and less reliable sources. By fact-checking the content they present, these portals can ensure that they are providing accurate and reliable information to their readers.

There are several fact-checking organizations that work to verify the accuracy of news stories, FactCheck.org, Snopes, and Full Fact. These organizations work independently of any news organization and provide fact-checking services to the public. They check the accuracy of statements made by politicians, public figures, and other news outlets, and provide their findings to the public. Fact-checking is an essential part of the journalistic process and helps ensure that the public is receiving accurate information. It is especially important in today's digital age, where misinformation can spread quickly and easily.

Scientific data and research from Harvard Kennedy school have provided insights into the effectiveness of fact-checking organizations and their impact on the public, see[8]. For example, research has shown a significant increase in the debunking efforts of fact-checking organizations in 2020, driven by fact-checking of COVID misinformation. User engagement with fact-checking content was found to be influenced by region-specific salient events and viral tweets. However, there was substantial heterogeneity across fact-checking organizations and regions in how engagement with coronavirus-related content correlated with overall engagement on fact-checks, see[9].

The role of fact-checking organizations has become increasingly important due to the widespread prevalence of misinformation. Fact-checking activity has been shown to have relevant effects on political beliefs. For instance, fact-checking was found to have an overall positive influence on political beliefs, but its ability to correct misinformation is significantly weakened by individuals' pre-existing beliefs, ideology, and knowledge. Moreover, debunking activity can have a deterrent effect on candidates, since it raises the reputational costs associated with lies, thus improving information accuracy

Pre-emptive fact-checking, often termed "pre-bunking," can impact individuals' ability to recognize the truth when exposed to fake news. Tags such as disputed or rated false make users perceive false headlines as less accurate. Furthermore, the correction of health-related misinformation was found to have positive effects, and these effects can be higher the more individuals are involved with topics covered by the fake content1.

However, there are also risks and drawbacks associated with fact-checking activity. For example, the repetition of either fake or real news tends to increase the perceived accuracy of that news. A detailed debunking message surprisingly positively correlates with the persistence of the misinformation effect. Moreover, the impact of fact-checking is seriously threatened by the different communities of fact-check sharers versus misinformation sharers, the short period of time in which fact-checks are likely to spread, and the amount of shared misinformation which is disproportionately higher than fact-checking content1.

In conclusion, fact-checking plays a critical role in the dissemination of news and information, particularly in the current digital age where misinformation can spread rapidly. Although it has its challenges, fact-checking serves as an important tool in promoting the accuracy of information and combating misinformation. It is important to continue supporting and improving fact-checking efforts to ensure the public has access to accurate and reliable information.

1.4 Bellingcat

Bellingcat is an independent international collective of researchers, investigators, and citizen journalists who use open-source and social media investigation to probe a variety of subjects. It was founded by British journalist and former blogger Eliot Higgins in July 2014, see[35]

Structure of Bellingcat is as follows. Bellingcat operates as a decentralized network of individuals located in various countries, who contribute their unique skills and expertise to investigations. The organization relies heavily on volunteers and crowd-sourced investigation, with a small core team coordinating the efforts. Bellingcat's work is primarily published on its website, which serves as a hub for its investigations and methodological guides.

To gather information Bellingcat uses open-source intelligence (OSINT) to gather information. OSINT refers to data collected from publicly available sources, such as social media, satellite imagery, and government reports. They use a variety of digital tools and techniques to verify and analyse this data, including geolocation, digital forensics, and reverse image searching.

In peace time, Bellingcat's investigations often focus on issues like corruption, organized crime, and environmental harm. They gather information from a wide range of sources, including local news reports, public records, and social media posts. During a war, Bellingcat's approach to information gathering changes. They may focus more on analysing satellite imagery to track troop movements, or on social media posts from conflict zones to document potential war crimes. They also pay close attention to disinformation campaigns and strive to debunk false narratives.

Bellingcat places a strong emphasis on verification and fact-checking. They use a variety of methods to verify the information they gather, such as cross-referencing sources, checking the reliability of sources, and using digital forensics to confirm the authenticity of images and videos. During a war, these methods become even more important, as the risk of disinformation increases.

1.5 Associated Press (AP)

The Associated Press (AP) as a representative news portal, see[36]. The AP is a non-profit organization that is not government-funded or sponsored by any corporations. It is often credited as the primary source of many stories that other outlets pick up and run for their own readers. It has a crowd-sourced bias rating from All Sides of "left-leaning," but this can fluctuate over time.

For news gathering during peacetime, reputable agencies like the AP utilize an extensive network of journalists around the world. These journalists collect information through multiple methods, including interviews, observations, and document retrieval. They then send their reports back to the central newsroom where editors review and fact-check the information before it is published.

In conflict or war zones, the process of gathering news becomes more complex due to the increased risks and challenges. One key initiative to help journalists in such situations is the training program on field safety in war zones offered by the Peace Journalism Lab (PJL) at Aristotle University Thessaloniki see[10]. This program equips journalists with the necessary knowledge to cope with the challenges they are likely to face in conflict areas, including personal security, data security, and health and stress management. The program also promotes peace journalism, which seeks to use journalism to advance conflict resolution and emphasizes reporting on the roots of conflict and highlighting possible solutions.

1.6 Reuters

Reuters is a prominent international news agency headquartered in London see[37], known for its comprehensive news coverage, including financial, business, political, and general news. It is one of the oldest news agencies globally, established in 1851, with a reputation for integrity, independence, and freedom from bias1.

During peacetime, Reuters operates a vast network of journalists and reporters stationed around the world, covering various news stories ranging from politics to financial markets, human interest stories, and more. For instance, I found an article about the 2023 UK general election, showing the kind of political coverage Reuters undertakes during peacetime, see[20].

During conflict, Reuters continues its on-the-ground reporting, as indicated by a detailed report on the Yemen conflict. The article gives an in-depth view of the situation, indicating that Reuters maintains an active presence in conflict zones to provide real-time updates and accurate information to its readers. This demonstrates Reuters' commitment to reporting on significant global events, even in challenging and dangerous environments.

In times of conflict, Reuters faces unique challenges in ensuring the safety and security of its journalists while delivering accurate and unbiased news. Conflict zones often present heightened risks, including physical dangers, restrictions on media access, and the potential for misinformation and propaganda. To overcome these challenges, news agencies like Reuters employ stringent safety protocols, rely on local reporters and fixers with in-depth knowledge of the region, and maintain close relationships with international organizations and security forces to navigate complex situations.

The Yemen conflict serves as an example of Reuters' dedication to providing comprehensive coverage during ongoing hostilities. By having journalists on the ground, Reuters can gather first-hand accounts, witness events as they unfold, and provide valuable insights into the complexities of the conflict. This type of reporting helps readers around the world understand the humanitarian crisis, political dynamics, and the impact on civilian populations.

Reuters' reputation for integrity, independence, and freedom from bias is crucial during times of conflict. As misinformation and propaganda often proliferate during war, readers rely on trusted news sources to provide accurate and balanced information. Reuters' commitment to adhering to high journalistic standards helps ensure that its reporting maintains credibility and helps readers make informed judgments about the situation.

In summary, Reuters continues its comprehensive news coverage during both peacetime and conflict, employing a global network of journalists to report on a wide range of topics. Its presence in conflict zones underscores its commitment to delivering accurate, real-time information to its readers, even in challenging and dangerous environments. By upholding its reputation for integrity and independence, Reuters plays a crucial role in providing reliable news during times of conflict.

1.7 Comparison Bellingcat, the Associated Press (AP), and Reuters

The differences and similarities of Bellingcat, the Associated Press (AP), and Reuters and their approaches to gathering information we have investigated, and the results are below.

Nature and Structure:

- Bellingcat operates as a decentralized collective, relying on volunteers and crowd-sourced investigation, while the AP and Reuters are established news organizations with structured hierarchies.

- Bellingcat focuses on open-source investigation, while the AP and Reuters employ a broader range of journalistic methods, including interviews and document retrieval.
- Bellingcat specializes in investigations using OSINT, while the AP and Reuters cover a wide range of news topics.

Focus and Expertise:

- Bellingcat often delves into issues like corruption, organized crime, and environmental harm, while the AP and Reuters cover a broader range of topics, including politics, business, and general news.
- Bellingcat excels in analysing digital content and verifying open-source information, while the AP and Reuters have extensive networks of journalists for on-the-ground reporting.

Independence and Ownership:

- Bellingcat is an independent collective, driven by a network of contributors, while the AP is a non-profit organization, and Reuters is a prominent international news agency.

Similarities:

- Bellingcat, the AP, and Reuters all emphasize the importance of verification and fact-checking in their reporting.
- They employ various methods, such as cross-referencing sources, digital forensics, and relying on trusted networks, to ensure the accuracy of their information.
- Coverage During Conflict:
  Bellingcat, the AP, and Reuters maintain a presence in conflict zones and provide real-time updates to their readers.

They face unique challenges in gathering news in dangerous environments, but they strive to deliver accurate information amid potential risks and misinformation.

Commitment to Integrity:

Bellingcat, the AP, and Reuters have reputations for integrity, independence, and freedom from bias, they adhere to high journalistic standards to provide reliable and unbiased news to their audiences.

While Bellingcat, the AP, and Reuters have different structures, focuses, and approaches to news gathering, they all share a commitment to verification, coverage during conflict, and maintaining integrity. Bellingcat's strength lies in open-source investigation, while the AP and Reuters leverage their extensive networks for diverse news coverage. The AP and Reuters have long-established reputations as news organizations, while Bellingcat operates as a more decentralized collective. Collectively, these organizations contribute to the broader landscape of journalism, each bringing unique perspectives and methods to inform the public and navigate the challenges of reporting in various contexts.

1.8 Research questions

To guide our investigation, the following research questions will be addressed:

- How does an internet news portal function, gather information, and detect fake news during war situations?
- How can machine learning tools be leveraged within the internet news portal to enhance the detection of fake news during war situations?
- Can improvements be proposed for the organization's system, specifically the internet news portal, to enhance fake news detection during war situations?

By answering these research questions, this study aims to contribute to the understanding of information management in the context of war and provide insights into the application of machine learning tools for fake news detection within an internet news portal. The findings will not only benefit researchers and practitioners in the fields of journalism, media studies, and data science but also inform the development of strategies and policies to combat the spread of fake news during war situations.

1.9 Thesis Structure

This thesis is structured as follows:

- Chapter 1, Introduction provides an overview of the research topic, motivation, research questions, and the thesis structure.

  In Chapter 1, titled "Introduction," we provide an overview of the research topic and its significance. We explore different types of news agencies, including internet news portals, and delve into the concept of fact-checking. Furthermore, we introduce Bellingcat, the Associated Press (AP), and Reuters as specific organizations for comparison. In this chapter, we also present the research questions that guide the study. Finally, we outline the structure of the thesis to give readers a roadmap of the subsequent chapters.
- In Chapter 2, "Fake News Types," we delve into the context, motivations, targets and consequences, techniques, and content, as well as amplification and dissemination of fake news. This comprehensive exploration in Chapter 2 provides a foundation for the subsequent chapters.
- Next, we move on to Chapter 3, "Literature Review," where we conduct a thorough review of existing research on fake news. This chapter focuses on motivations, content and themes, consequences, and impact, as well as amplification and dissemination. By synthesizing previous studies, we identify gaps in the current knowledge and highlight the significance of our research in addressing these gaps.
- The methodology chapter, Chapter 4, "Methodology," outlines the research approach and methods used in this study. We describe the data cleaning process and introduce the datasets, including the Syrian war fake news dataset and the fake news datasets during normal times and the Iraq war. This chapter provides insights into the selection and preparation of the data for analysis.
- In Chapter 5, "Vectorization Techniques," we explore various vectorization techniques used to transform textual data into numerical representations. We discuss TF-IDF vectorization, Count Vectorizer, and Hashing Vectorizer, outlining their purposes, advantages, and limitations. This chapter provides the necessary background for understanding the feature extraction process in fake news detection.
- Moving forward, Chapter 6, "Artificial Intelligence Tool for Detecting Fake News," delves into the artificial intelligence tools used in fake news detection. This chapter covers the basics of artificial intelligence and machine learning, followed by a detailed exploration of natural language processing techniques. We introduce classifiers such as the Passive Aggressive Classifier, Logistic Regression, SGDClassifier, Convolutional Neural Network (CNN), and Deep Neural Network (DNN). By understanding these tools, readers gain insights into the techniques utilized in fake news detection.
- The experimental setup and evaluation chapter, Chapter 7, "Experimental Setup and Evaluation," explains the procedures and metrics used to evaluate the performance of the

fake news detection models. We discuss the train-test split, evaluation metrics, cross-validation, parameter configurations, and the rationale behind these choices. This chapter provides transparency and clarity regarding the experimental methodology.

- In Chapter 8, "Results and Analysis," we present the findings obtained from the experiments and provide a detailed analysis of the performance of the classifiers and vectorizers. We explore the implications of these results, including the enhanced dataset, domain-specific feature engineering, ensemble methods, deep learning models, real-time data updates, and continuous monitoring and evaluation. This chapter allows readers to understand the significance of the research findings and their implications for fake news detection.
- Finally, the "Conclusion" chapter summarizes the main findings of the thesis and provides a comprehensive overview of the research. It reflects on the research questions, discusses the contributions to the field, practical applications, and future directions. This concluding chapter offers a holistic view of the research journey and emphasizes the importance of the study in advancing the field of fake news detection.

By following this structured approach, the thesis aims to provide a thorough examination of fake news detection, incorporating a range of techniques, methodologies, and analyses to contribute to the understanding and mitigation of the fake news phenomenon.

2. Fake news types

Understanding the multifaceted landscape of fake news is essential for effectively identifying and combating it, see[28], [31] and [30]. There are several common categories of fake news, each with its own unique characteristics and impacts:

- **Fabricated News:** This form of fake news is entirely fictional and has no grounding. Designed to intentionally deceive and mislead the public, fabricated news pieces often create entirely false scenarios or events. For instance, during the 2016 U.S. Presidential Election, a fabricated news story falsely claimed that a Washington D.C. pizzeria was a front for a child trafficking ring involving high-ranking Democrats.
- **Manipulated News:** As the name suggests, manipulated news uses real events or facts as its basis but distorts them in a way that misleads or deceives the audience. Examples of manipulation can range from altering images or videos, taking statements out of context, or misquoting sources to present a false narrative. The infamous "Nancy Pelosi Slurring Her Words" video, which was slowed down to make it appear as if she was intoxicated, is a pertinent example of this type of fake news.
- **Clickbait:** Clickbait is a form of sensationalism where exaggerated headlines or claims are used to entice people into clicking on a link. This type of fake news prioritizes web traffic and revenue generation over the dissemination of accurate information. An example is a headline that promises a shocking revelation or scandal, but the linked content does not deliver on the promise.
- **Propaganda:** Propaganda is used to sway public opinion or promote a specific agenda, often used by governments, political parties, or interest groups. It manipulates people's perceptions through the spread of misinformation, often in a systematic and strategic manner. The Nazi propaganda machine during World War II is one of the most infamous examples of this type of fake news.
- **Hoaxes:** Unlike other types of fake news, hoaxes are typically spread as pranks or jokes without an intent to deceive. However, if taken seriously, they can lead to serious

consequences. An example of this is the "War of the Worlds" radio broadcast in 1938, which led some listeners to believe an actual alien invasion was occurring. With a firm grasp of the different types of fake news, readers can become more discerning and better equipped to identify and avoid false information. However, the realm of fake news takes on a different complexion during times of war, as the nature of the conflict and the motivations of the involved parties introduce new forms of misinformation:

- **Propaganda:** Governments and military organizations often conduct propaganda campaigns during war to shape public opinion and further their agendas. For instance, during the Vietnam War, both the U.S. and Vietnamese governments used propaganda to control the narrative of the war.
  Misinformation about military operations: Misinformation related to military operations aims to deceive the enemy and mislead the public. For example, during World War II, Operation Fortitude was a successful misinformation campaign that convinced the Germans that the D-Day invasion would occur at Pas de Calais, not Normandy.
- **False casualty reports:** To manipulate perceptions of a war's progress or the strength of the opposing forces, false reports about the number of casualties can be circulated. One example is during the Iraq War, where casualty numbers were often disputed and used for propaganda purposes.
- **Atrocities and war crimes:** False reports of atrocities or war crimes can be used to stoke outrage and support for military actions. During the Yugoslav Wars, both sides often accused the other of committing atrocities, leading to increased animosity and continued conflict.
- **Fake humanitarian appeals:** Fake appeals for humanitarian aid or assistance can be used to exploit sympathy from the international community or misrepresent the scale of suffering. For instance, during the Syrian Civil War, there were numerous instances of fake appeals being circulated on social media. While the types of fake news can vary depending on the specific conflict, geopolitical factors, and the technological landscape at the time, the goal is to manipulate information, control narratives, and shape public perception to gain an advantage in the war effort.

The distinction between fake news in a general context and that which transpires during times of war lies in the specific context, motivations, and consequences tied to armed conflicts. Several pivotal distinctions can be drawn below.

2.1 Fake news creation

Fake news refers to deliberately false or misleading information presented as news, see [30]. It can be created using various techniques, including selective editing, fabrication of events or people, and machine-generated content. Let us explore these aspects in more detail:

- **Selective editing:**
  Fake news articles can be produced by selectively editing information. This involves taking genuine news or information and manipulating it by altering details such as names, dates, statistics, or quotes. By changing these elements, the overall narrative can be distorted or completely misrepresented, leading to misinformation.
- **Fabrication of events or people:**
  Another method involves completely fabricating events or people that never occurred or existed. This can include inventing stories, creating fictional sources, or falsely

attributing statements to individuals or organizations. Such fabrications are designed to mislead readers and create a false narrative.

- **Machine-generated fake news:**
  Advances in artificial intelligence and natural language processing have enabled the creation of machine-generated fake news. With the help of AI algorithms, it is becoming increasingly easy to generate text, images, and even videos that mimic human-like content. These algorithms can learn from vast amounts of existing data to produce highly convincing but entirely false information. This poses a significant challenge, as AI-generated fake news can spread rapidly and deceive unsuspecting readers.

Machine-generated fake news often relies on generative models, such as language models, to generate text that closely resembles human-written content. These models can be fine-tuned or trained on specific datasets to mimic the style, tone, and structure of legitimate news articles. However, instead of providing accurate and reliable information, the generated content aims to manipulate or deceive the audience.

To combat the spread of fake news, it is essential to foster media literacy, critical thinking, and fact-checking skills among individuals. Additionally, there are ongoing efforts to develop automated tools and algorithms to detect and flag potential fake news sources. Fact-checking organizations and news outlets play a vital role in verifying information and debunking false narratives.

Addressing the challenge of machine-generated fake news requires a combination of technological advancements, ethical guidelines, and collaborative efforts among researchers, policymakers, and platforms hosting online content.

2.2 Motivations of creating fake news

Motivations behind ordinary fake news can be diverse, ranging from financial gain to ideological promotion. However, fake news in times of war serves specific military or political objectives. It is used as a calculated tool to shape public opinion, maintain morale, manipulate enemy perception, or garner international support. The primary goal extends beyond personal or ideological benefits; it aims to secure an advantage in the war effort. A historical example can be seen in the propaganda disseminated by the U.S. and the U.S.S.R during the Vietnam War and the Afghan War, respectively, with each side seeking to tilt global opinion in their favour, see[38]

2.3 Targets and Consequences

War-related fake news targets a broad spectrum of audiences. It aims to influence not only the civilian population but also military personnel, governments, and international actors. The repercussions of such fake news can be profound, leading to loss of life, escalation of the conflict, erosion of trust, and prolonged suffering. In comparison, ordinary fake news primarily impacts public discourse, individual beliefs, or reputation, with no immediate life-threatening consequences. For example, during the Rwandan genocide, radio broadcasts spread misinformation that incited violence, leading to mass atrocities, see[39]

2.4 Techniques and Content

Fake news during times of war often exhibits unique techniques and content that are tailored to the conflict. Themes of propaganda, misinformation about military operations, false casualty reports, and appeals for humanitarian aid are prevalent. This type of fake news relies on a concoction of fabricated information, distorted facts, and manipulation of emotions to achieve its objectives. For example, during the Gulf War, reports of babies being taken out of incubators by Iraqi soldiers were later proven to be false see [40]. In contrast, normal fake news spans a broad array of topics, from politics and health to entertainment and social issues.

2.5 Amplification and Dissemination

Amplification and dissemination of fake news during times of war can be particularly intense and urgent. Various communication channels, including traditional media, social media platforms, state-controlled outlets, and covert networks, are utilized to spread false narratives and manipulate public opinion.

During armed conflicts, the parties involved often exploit digital platforms to rapidly disseminate their narratives and gain international support. An example of this is the 2008 Russia-Georgia War see [41], where both Russia and Georgia used online platforms to amplify their versions of events and advance their agendas. The speed and reach of social media make it an attractive tool for spreading fake news during conflicts, as information can be shared widely and rapidly without thorough fact-checking.

Detecting and combatting fake news in times of war require specialized strategies and tools. Models and algorithms used for fake news detection need to be trained on data specific to armed conflicts to understand the nuances and patterns of war-related misinformation. This may include analysing historical data from previous conflicts, monitoring current events in real-time, and studying the tactics and techniques employed by conflicting parties to spread fake news.

Collaboration between governments, organizations, and technology platforms is essential to minimize the harmful effects of war-related fake news. By working together, these stakeholders can develop and implement measures to counter misinformation effectively. For instance, technology platforms like Facebook have collaborated with fact-checkers and third-party auditors during conflicts like the 2020 Nagorno-Karabakh conflict, see[42]. This partnership played a crucial role in identifying and flagging false information, reducing the spread of misinformation on the platform, and providing users with more accurate information.

It is also important to promote media literacy and critical thinking among the public, equipping them with the skills to evaluate information critically, discern reliable sources, and identify potential fake news. Educational initiatives and awareness campaigns can help individuals navigate the complex information landscape during times of war and make informed judgments about the veracity of news and events.

Overall, addressing the challenges posed by fake news in times of war requires a multi-faceted approach that combines specialized detection strategies, collaborative efforts among stakeholders, and the promotion of media literacy and critical thinking skills. By working together, we can mitigate the impact of fake news and foster a more informed society even during the tumultuous times of armed conflict.

3. Literature Review

Scholarly investigations have underscored the variances in the motivations, content, consequences, and dissemination strategies behind fake news during times of peace and war. These differences, rooted in the distinct contexts in which the misinformation is propagated, necessitate unique countermeasures and frameworks for detection. This literature review explores these differences in greater depth, drawing on a range of studies to elucidate the specific characteristics of war-related fake news, see [29], [30], [31].

3.1 Motivations and Objectives

Research points towards distinct motivations underlying the propagation of fake news during armed conflict, as opposed to times of peace. In the context of war, misinformation is often disseminated by state actors, military factions, or politically motivated entities, with the aim of achieving strategic military or political objectives (Fisher & Taub, 2020), see [33]. The principal aim is to shape public opinion, maintain morale among troops and citizens, manipulate enemy perception, or gain international support for their cause. A case in point is the disinformation campaign during the Russo-Ukrainian War, where both sides sought to control the narrative and influence international sentiment (Khaldarova & Pantti, 2016), see [21] and [32].

In contrast, the motivations driving the spread of normal fake news are typically more diverse and less politically charged. Misinformation in times of peace can stem from personal gain, such as generating revenue through increased website traffic, or from ideological beliefs, often polarizing societal discourse. Moreover, a desire for sensationalism often drives the creation and dissemination of such news, without immediate implications on war or peace (Tandoc et al., 2018).

3.2 Content and Themes:

The content and themes of fake news also vary significantly between war and peace times. According to an article by Reporter agency about war-related fake news often revolves around distinct themes reflective of the conflict context see[34]. Common motifs include propaganda, misinformation about military operations, false casualty reports, and appeals for humanitarian aid. These themes exploit the emotional intensity of war, the dynamics of military operations, and geopolitical contexts to manipulate narratives and perceptions.

In contrast, normal fake news spans a wide spectrum of topics, including politics, health, entertainment, or social issues, driven by diverse motives and agendas. A study by Pennycook and Rand (2020) shows that misinformation related to health, for instance, can spread rapidly in peaceful contexts, as evidenced by the Covid-19 pandemic, see [31]

3.3 Consequences and Impact:

The consequences and impact of war-related fake news often have far-reaching effects that transcend the realm of misinformation. Research conducted by Benkler et al. (2018), see[43] emphasizes that in times of conflict, the spread of false information can have severe repercussions, such as loss of life, escalation of violence, erosion of trust, and prolonged suffering. In contrast, the impacts of normal fake news, while influential in shaping public discourse, do not typically result in immediate life-and-death consequences. The unique stakes and sensitivities of war heighten the potential harm associated with the spread of misinformation, elevating it to a matter of national and international security.

3.4 Amplification and Dissemination:

The dissemination and amplification of war-related fake news involve specific communication channels and strategies tailored to the conflict context. State-controlled media, social media platforms, traditional media outlets, and covert networks play pivotal roles in propagating war-related misinformation (Bradshaw & Howard, 2019), see[44]. The urgency and intensity of war contribute to the rapid and widespread dissemination of fake news, potentially impacting military operations, international relations, and civilian perceptions.

In times of armed conflict, the propagation and amplification of fake news often involve coordinated efforts from state actors, military factions, or politically motivated entities. These entities strategically leverage various communication channels to disseminate their narratives and shape public opinion. Social media platforms offer an accessible and influential avenue for spreading war-related misinformation due to their wide reach and real-time nature. Compared to normal fake news, which can spread organically through social networks without centralized coordination (Vosoughi et al., 2018), see[29], war-related fake news exhibits a higher degree of orchestration. State-controlled media outlets and covert networks may also play significant roles in disseminating propaganda or manipulating information to advance specific military or political objectives (Khaldarova & Pantti, 2016), see[32].

The urgency and intensity of armed conflict create an environment conducive to the rapid dissemination of misinformation. The emotional intensity of war and the high stakes involved make war-related fake news particularly potent in influencing public perceptions and shaping the course of military operations.

By understanding the distinct amplification and dissemination strategies employed in times of war, researchers, policymakers, and technology platforms can develop targeted approaches to detect, mitigate, and counter the spread of war-related fake news. Collaborative efforts among governments, organizations, and technology platforms become imperative to minimize the harmful effects of such misinformation and preserve the integrity of information during times of conflict.

4. Methodology

4.1 Data cleaning

Data cleaning is a crucial step in any machine learning model see [23] and [18], particularly in the context of Natural Language Processing (NLP). Without proper cleaning, the dataset consists of unstructured text that computers cannot comprehend. In this section, we will outline the step-by-step process of data cleaning for an NLP project, using an article as a reference.

The data cleaning process for NLP typically involves the following steps:

Step 1: Punctuation Removal

Punctuation marks, such as periods, commas, question marks, and exclamation points, do not typically contribute meaningful information to NLP models see[18]. In the context of text analysis, they are often treated as noise that can interfere with the accurate understanding and interpretation of the text. Therefore, the first step in the data cleaning process for NLP is to remove punctuation from the text. To accomplish this, various techniques and libraries can be employed. In the provided article, the "string" library is utilized, which provides a set of thirty-two commonly used punctuation marks. These punctuation marks include symbols like '.', ',', '?', '!', and many others. By identifying and removing these punctuation marks, we can eliminate unnecessary noise and focus on the relevant textual content, see[18].

To implement the punctuation removal step, a function called "remove_punctuation" is created. This function facilitates the iteration over each word in the text and removes any punctuation marks encountered. The "string.punctuation" set, available in the "string" library, is used to define the set of punctuation marks to be removed. This set includes characters such as '.', ',', '!', '?', ';', ':', and more.By applying the "remove_punctuation" function to the text, we can strip away punctuation marks, resulting in a clean and punctuation-free version of the text. This processed text is then stored in a new column, often named "title_wo_punct" or something similar, to indicate that it contains the original text without any punctuation.

The removal of punctuation marks is an important initial step in data cleaning for NLP tasks. By eliminating punctuation, we ensure that the subsequent analysis and modelling focus on the meaningful words and phrases within the text, leading to more accurate and reliable results.

Overall, the goal of Step 1 is to transform the raw text data into a cleaner and more structured format, setting the foundation for further processing and analysis in subsequent steps of the data cleaning process.

Step 2: Tokenization

Tokenization is a fundamental step in the data cleaning process for NLP tasks see [18]. It involves breaking down longer texts, such as sentences, paragraphs, or entire documents, into smaller units called tokens. These tokens can be individual words, phrases, punctuation marks, or any other meaningful components that constitute the text.

The process of tokenization goes beyond simply separating the text into individual words or characters. It also considers the context and linguistic nuances of the text. For instance, when

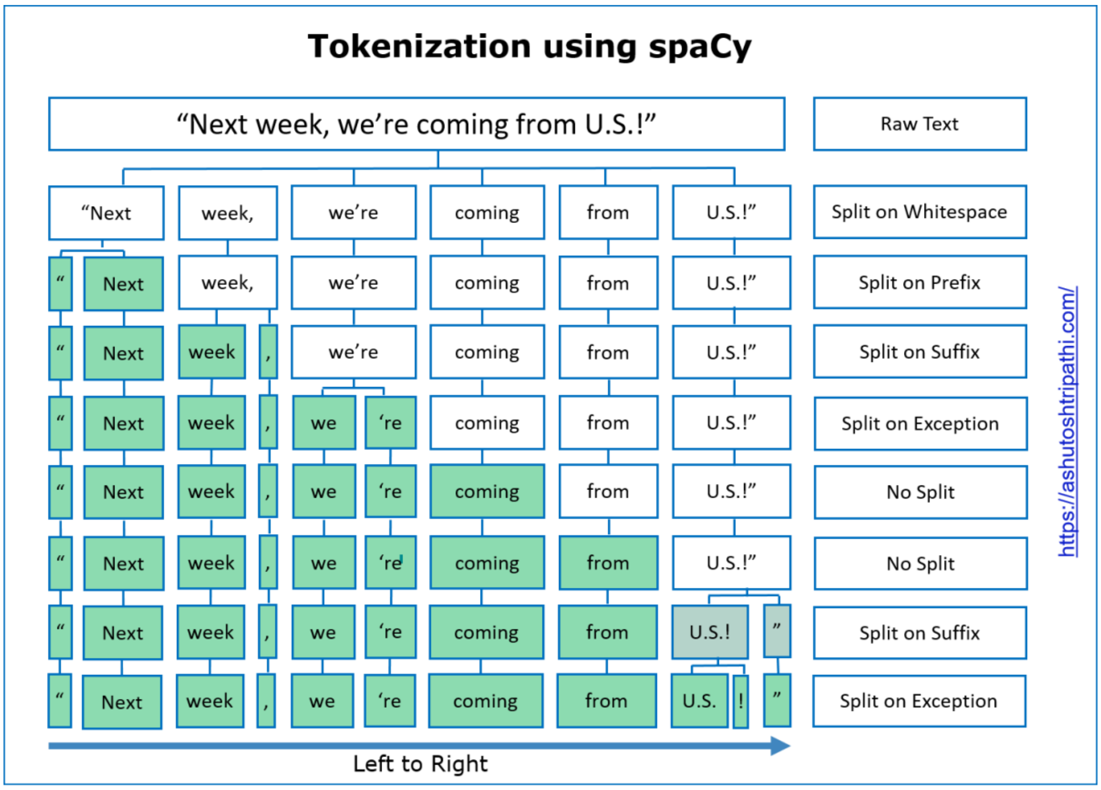


FIGURE 1 TOKENIZATION, SOURCE [18]

encountering punctuation marks, tokenization distinguishes between cases where the punctuation is a standalone token and cases where it is part of an abbreviation or an acronym.

The spaCy library is utilized for tokenization. spaCy employs an intelligent tokenizer that not only splits the text into tokens but also takes into consideration the specific context in which punctuation marks appear. For example, if a period (".") is encountered, the tokenizer determines whether it is a punctuation mark indicating the end of a sentence or if it is part of an abbreviation like "U.S." In the latter case, the tokenizer recognizes that the period should not separate the "U.S." into separate tokens.

This intelligent tokenization process helps to maintain the integrity of meaningful phrases, abbreviations, or acronyms, preserving their intended context and avoiding unnecessary fragmentation. By accurately identifying and separating tokens, we create a more granular representation of the text, enabling subsequent analysis and modelling techniques to capture the finer details and nuances present in the data.

By applying tokenization, we transform the original text into a structured collection of tokens. Each token represents a discrete unit of information that can be further analysed, processed, or used in subsequent NLP tasks. The resulting tokenized text facilitates tasks such as text classification, sentiment analysis, or information retrieval, where the focus is on individual elements rather than the entire text.

Step 3: Stop Word Removal

Stop words are commonly used words in a language that have little or no significance in determining the overall meaning of a text see [23] and [18]. Examples of stop words in English include "the," "is," "and," "of," and so on. These words appear frequently in texts but often carry little semantic value. Removing stop words from the dataset can help reduce noise and improve the quality of analysis.

In the provided article on wikidocs “Complete Guide to Spacy Tokenizer” the data cleaning process includes the removal of stop words. The nltk library (Natural Language Toolkit) is utilized to access a predefined list of English stop words. The library provides a comprehensive set of commonly used words that can be considered as stop words in NLP tasks. To remove the stop words, a function called "remove_stopwords" is created. This function iterates over each word in the list of words and checks if it matches any of the stop words from the nltk stopword list. If a word is found to be a stop word, it is removed from the list of words. The resulting list of words without stop words is then stored in a new column called "title_wo_punct_split_wo_stopwords."

The removal of stop words is beneficial for several reasons. First, it helps to eliminate noise in the dataset by discarding words that are commonly used but provide limited contextual or semantic information. By removing these words, the focus is shifted to more meaningful and informative words, which can enhance the accuracy and effectiveness of subsequent analysis and modelling tasks.

Furthermore, removing stop words can help reduce the dimensionality of the data. Since stop words are present in many texts and do not contribute significantly to the overall meaning, their removal can lead to a more concise and representative representation of the text. This reduction in dimensionality can improve computational efficiency and reduce the risk of overfitting in machine learning models.

However, it is important to note that the removal of stop words is not always necessary or beneficial in all NLP tasks. In certain cases, such as sentiment analysis or topic modeling, stop words may carry important contextual information and removing them could result in the loss of crucial insights. Therefore, the decision to remove stop words should be based on the specific task and the characteristics of the dataset.

Step 4: Lemmatization or Stemming

Lemmatization and stemming are techniques used to reduce words to their base or root forms, see[7]. These techniques help to normalize words, reduce the number of unique words in the dataset, and improve the efficiency and accuracy of subsequent analysis and modelling tasks.

Lemmatization considers the context and meaning of a word to determine its base or dictionary form. It considers the part of speech (POS) of the word and applies morphological analysis to produce the root form. For example, the lemma of the word "running" would be "run," as it represents the base form of the word. Lemmatization ensures that the resulting word is a valid word that can be found in a dictionary or linguistic resource.

On the other hand, stemming is a more simplistic technique that chops off the end of a word without considering the context or meaning. Stemming relies on a set of predefined rules to remove common suffixes or prefixes from words. For example, stemming the word "running" would result in "run," but it does not take into account the fact that "run" has different meanings

based on its usage as a verb, noun, or adjective. Stemming may produce root forms that are not actual words but still serve the purpose of reducing words to their common base form.
In the provided article, both lemmatization and stemming techniques are used to reduce words to their root forms. The nltk library (Natural Language Toolkit) is used for lemmatization, while the stemming library is used for stemming. These libraries provide predefined rules and resources to perform these techniques efficiently.

To illustrate the differences between lemmatization and stemming, the article provides an example using the word "believe." Lemmatization would reduce it to its base form "believe" because the context and meaning of the word are considered. In contrast, stemming would simply remove the suffix and reduce it to "believe" without considering the context or meaning. The resulting lemmatized or stemmed words are then compared and stored in a new column, capturing the normalized representation of the original words.

The main goal of lemmatization and stemming is to reduce the number of unique words in the dataset. By converting different inflected forms of a word to a common base form, the focus is directed towards the essential semantic content of the text rather than its superficial variations. This can lead to improved text analysis, as it helps to consolidate the occurrence of related words and allows for better identification of patterns and relationships.
However, it is important to note that the choice between lemmatization and stemming depends on the specific requirements of the task and the characteristics of the dataset. Lemmatization is generally preferred when maintaining the semantic meaning and grammatical correctness of words is crucial. On the other hand, stemming may be more suitable in cases where simplicity

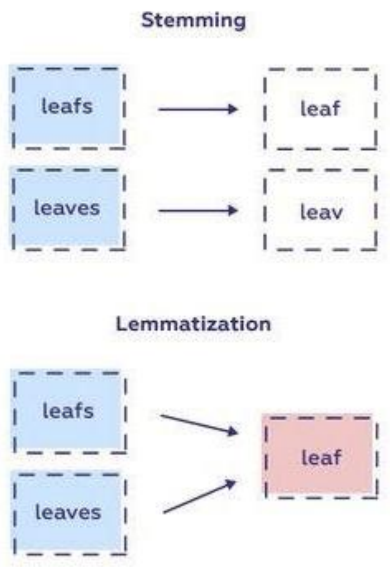


FIGURE 2 LEMMATIZATION OR STEMMING,SOURCE[7]

and efficiency are prioritized, and the semantic nuances of words are less critical.
Depending on the specific dataset and project requirements, additional cleaning steps can be performed to further refine the text data. These steps can be customized based on the nature of the dataset and the desired quality of the cleaned text.
Step 5: Removal of URLs.
Text data from various sources see [18], such as social media or web scraping, may contain URLs that are not relevant to the analysis. Removing URLs is a common preprocessing step to eliminate unnecessary information from the text. This can be achieved using regular expressions or string manipulation techniques to identify and remove URLs from the text.

Step 6: Removal of HTML tags.
If the dataset contains HTML or XML documents, the text may be embedded within HTML tags. In such cases, it is necessary to remove these HTML tags to extract the plain text. Libraries like BeautifulSoup can be utilized to parse and clean the HTML or XML documents, ensuring that only the textual content remains for further analysis see [18].
Step 7: Removal of Emojis.
Emojis are graphical representations often used in text communication to convey emotions or sentiments. However, for some NLP tasks, emojis may not contribute relevant information and can be considered noise in the dataset. Removing emojis involves identifying and removing these graphical symbols from the text, ensuring that only the textual content remains.
Step 8: Removal of Numbers
Numeric characters or numbers may not be relevant to the analysis or the specific NLP task at hand. Removing numbers involves identifying and excluding numeric characters from the text. This can be achieved using regular expressions or string manipulation techniques to identify and remove numeric values, ensuring that only textual information is retained. By following these additional cleaning steps, the dataset undergoes a thorough cleaning process, removing unnecessary content and focusing on the key attributes suitable for the machine learning model. These cleaning steps help improve the quality and relevance of the text data, enabling more accurate analysis, feature engineering, and model training.
It is important to note that the specific implementation details may vary depending on the programming language, libraries, and tools used for NLP. In Python, which is the environment commonly used for NLP tasks, several libraries are available to facilitate data cleaning see [18]:

- **Pandas:** Pandas is a powerful data manipulation library that provides data structures and functions for handling and analysing structured data. It is often used for loading, pre-processing, and manipulating textual data.
- **NLTK (Natural Language Toolkit):** NLTK is a comprehensive library for NLP tasks. It offers various functionalities such as tokenization, stemming, lemmatization, stop word removal, and more. NLTK also provides corpora, lexical resources, and pre-trained models for NLP tasks.
- **Spacy:** Spacy is a popular library for advanced NLP tasks. It offers efficient tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and other linguistic annotations. Spacy provides pre-trained models for different languages, making it a convenient choice for NLP projects.
- **Regular Expressions (re):** The "re" module in Python provides support for regular expressions, which are powerful tools for pattern matching and text manipulation. Regular expressions can be used for tasks such as tokenization, pattern matching, and extracting specific information from text.
- **BeautifulSoup:** BeautifulSoup is a library used for web scraping and parsing HTML or XML documents. It helps in extracting text and cleaning HTML tags from web pages, which can be useful when dealing with textual data from web sources.
- **TextBlob**: TextBlob is a library built on top of NLTK and provides an easy-to-use interface for common NLP tasks such as sentiment analysis, part-of-speech tagging, noun phrase extraction, and more. It also includes a simplified API for performing text cleaning and normalization.

These libraries offer a wide range of functionalities to support various cleaning tasks in NLP projects. Depending on the specific requirements and characteristics of the dataset, you can choose the appropriate library and utilize its functionalities to perform the necessary cleaning steps.

4.2 Datasets

4.2.1 Syrian war fake news

In the thesis we utilized a specific dataset called "Syrian war fake news Dataset." see [1].
This dataset was curated by researchers at the American University of Beirut (AUB) and focuses on the Syrian war, which adds a unique dimension to the available fake news datasets. While most existing datasets predominantly revolve around US politics, entertainment news, or satire, this dataset specifically explores news articles related to the Syrian war.
The dataset comprises news articles sourced from various media outlets, including mobilization press, loyalist press, and diverse print media. Each article in the dataset is labelled either as 0 (fake) or 1 (credible) based on its credibility assessment derived from a ground truth information obtained from the Syrian Violations Documentation Centre (VDC).
By incorporating this dataset into my research, I aimed to analyse the detection of fake news within the context of a war situation. The unique nature of news reporting during wartime and the scarcity of available sources for manually labelled news articles make this dataset a valuable resource for studying fake news in such conditions. Utilizing this dataset, I conducted comprehensive analyses and employed various techniques to detect and classify fake news articles accurately. The findings and insights gained from this study contribute to a better understanding of the challenges and implications of managing information and combating misinformation in organizations operating in war-affected regions.
Overall, the inclusion of the "Syrian war fake news Dataset" in my research provides a novel perspective and adds depth to the exploration of fake news detection within the complex dynamics of a war situation.

4.2.2 Fake news during normal times

The dataset downloaded from a website called "data-flair" containing 7,796 instances, see [2], with each instance consisting of four columns. The first column serves as an identifier for the news, while the second and third columns contain the title and text of the news articles, respectively. The fourth column plays a crucial role as it contains labels indicating whether the news is classified as REAL or FAKE. With a dataset size of 29.2MB, this collection of news articles serves as a valuable resource for investigating the detection and classification of fake news in normal circumstances. By analysing this dataset, I aimed to explore the effectiveness of various techniques and algorithms in identifying fake news during periods without the added complexity of war or conflict.

Through the application of machine learning and natural language processing methods to this dataset, I conducted extensive experiments and evaluations to develop robust models capable of accurately distinguishing between real and fake news articles. The insights gained from this analysis contribute to the development of effective strategies and tools for managing and combating fake news during non-crisis situations.
The utilization of the "Fake news during normal times - 1" dataset in my research enhances the understanding of fake news detection beyond war scenarios. It offers valuable insights into the prevalence and characteristics of fake news during periods of relative stability, enabling organizations to strengthen their information management systems and mitigate the impact of misinformation.

The second part of it is collected from Kaggle, see [3], the dataset includes columns such as "title," "text," and "label." The "title" column represents the title or headline of each article, providing a summary of the content. The "text" column contains the main body of the news article, providing detailed information and context. The "label" column specifies whether the news article is classified as real or fake. By examining the content of the first few rows, we can observe that the dataset covers various topics and subjects. The articles address subjects such as political figures, global events, and social issues. Some articles discuss specific incidents, while others provide analysis or opinion pieces.

For instance, the first row mentions Hillary Clinton's alleged fear, with the article written by Daniel Greenfield. This could potentially be categorized as a fake news article. The second row describes a moment involving Paul Ryan, but it is difficult to ascertain the credibility of the news based on the given information. The third row mentions John F. Kerry's visit to Paris as a gesture of sympathy, which is a real news article.

As we proceed through the dataset, we encounter news articles discussing various topics, including terrorism, international relations, and social events. Some articles contain subjective content or questionable claims, suggesting a potential inclination toward fake news. However, without examining the entire dataset, it is challenging to determine the overall distribution and characteristics of real and fake news articles.

In summary, the "Fake news during normal times - 2" dataset comprises news articles with titles, textual content, and corresponding labels indicating their classification as real or fake. The dataset covers a range of subjects, and the content of the articles suggests a mix of both real and potentially fake news.

4.2.3 Iraq war fake news.

Researchers at the Fund for Independence in Journalism sought to document every public statement made by eight top Bush administration officials from September 11, 2001, to September 11, 2003, see[4], regarding (1) Iraq’s possession of weapons of mass destruction and (2) Iraq’s links to Al Qaeda. Although both had been frequently cited as rationales for the U.S. war in Iraq, by 2005 it was known that these assertions had not, in fact, been true.

The centrepiece of this project was an exhaustive, searchable, and robustly indexed database of all public statements on the two topics by President George W. Bush, Vice President Dick Cheney, Secretary of State Colin Powell, National Security Adviser Condoleezza Rice, Defence Secretary Donald Rumsfeld, Deputy Defence Secretary Paul Wolfowitz, and White House Press Secretaries Ari Fleischer and Scott McClellan. These statements were painstakingly collected from the websites of the White House, State Department, and Defence Department as well as from transcripts of interviews and briefings, texts of speeches and testimony, prepared statements, and the like. Also included were statements in the same two categories that appeared in major newspapers and on television programs, were part of public statements by other officials, or were contained in government studies or reports, books, and the like from September 11, 2001, to December 31, 2007. Secondary material from reports and books was included in the two-year database only in cases where specific dates were available. Other noteworthy material was included for context and completeness.

As a general rule, only the relevant excerpts of public statements have been included in the database; deleted material is marked “[text omitted].” (In a case of a lengthy press conference in which Iraq is mentioned only briefly, for example, only the relevant passage is included.) Where deleting text might have rendered the remaining material misleading or difficult to understand, longer passages were left intact. And in some cases public pronouncements of Bush administration officials that did not include direct statements were included if they provided useful context.

5. Vectorization Techniques

In our research, the process of transforming text data into numerical representations is essential for effective analysis, see [23], [24] and [25]. To accomplish this, we utilized various vectorization techniques, namely TF-IDF Vectorization, CountVectorizer, and HashingVectorizer. In this section, we will provide an in-depth explanation of each technique, including their purpose, advantages, limitations, and the rationale behind their selection for our research.

5.1 TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization is a widely used technique in natural language processing. Its main objective is to assign weights to words based on their frequency within a particular document and across the entire corpus. By doing so, TF-IDF aims to capture the importance of words in distinguishing one document from another.

5.1.1 Purpose

TF-IDF Vectorization serves as a powerful tool in text analysis and information retrieval tasks. It goes beyond simply describing word frequency and aims to uncover the significance of words within a given context. By assigning weights to words based on their occurrence, TF-IDF provides a quantitative measure of their importance. This enables researchers to identify key terms that contribute significantly to the overall meaning and context of a document.

5.1.2 Advantages

Importance Weighting: TF-IDF allows the identification of words that are more representative and discriminative within a document. By considering both term frequency and inverse document frequency, it provides a nuanced understanding of word importance.

Feature Selection: TF-IDF aids in feature selection by assigning higher weights to informative words and lower weights to common words. This process highlights terms that contribute more significantly to the overall meaning and context of the document, facilitating more effective analysis and modelling.

Contextual Understanding: By considering the frequency of words within documents and across the corpus, TF-IDF provides a contextual understanding of their importance. This awareness enhances the accuracy and effectiveness of subsequent analysis and modelling techniques.

5.1.3 Limitations

Lack of Semantic Understanding: TF-IDF primarily relies on statistical measures and does not incorporate semantic understanding or contextual meaning. Therefore, it may not capture complete semantic relationships between words, leading to limitations in certain applications.

Vocabulary Limitations: TF-IDF relies on the vocabulary present in the corpus. If a relevant word is missing from the vocabulary, it will not contribute to the vector representation. This highlights the importance of thorough pre-processing and ensuring a comprehensive vocabulary for accurate vectorization. Given its established effectiveness and widespread adoption in natural language processing, TF-IDF vectorization was a logical choice for our research. By leveraging its ability to assign weights based on term frequency and inverse document frequency, we could focus on informative words while filtering out common and less meaningful terms. This technique has shown promising results in various text classification, information retrieval, and sentiment analysis tasks, making it a valuable addition to our research methodology.

Continued research has further expanded the boundaries of TF-IDF vectorization, introducing variations such as sublinear TF-IDF, TF-IDF with n-grams, and TF-IDF with document length normalization. These advancements enhance the technique's applicability and performance in a wide range of text analysis scenarios.

5.2 CountVectorizer

In our research, we utilized a vectorization technique called CountVectorizer. This technique plays a crucial role in transforming text documents into a format that can be processed and analysed by computers. It involves converting words and sentences into numerical representations, enabling us to apply mathematical and statistical techniques to understand and extract meaningful insights from the text data.

5.2.1 Purpose

CountVectorizer serves the purpose of counting the occurrences of words within a given set of documents. It scans the documents and creates a matrix that represents the frequency of each word. This matrix provides a basic yet informative representation of the text corpus, allowing us to analyse and draw conclusions based on word frequencies.

5.2.2 Advantages

Simplicity: CountVectorizer offers a straightforward and easy-to-understand approach to vectorization. It does not involve complex mathematical calculations or weighting schemes. Instead, it focuses on counting the occurrences of words, making it accessible for researchers and practitioners without extensive technical knowledge.

Preserving Word Frequency Information: One advantage of CountVectorizer is its ability to maintain the information about the frequency of words in the documents. This can be valuable in certain analysis scenarios where the raw count of words is of interest. By preserving this information, we can gain insights into the distribution and prevalence of specific terms in the text corpus.

5.2.3 Limitations

Lack of Importance Weighting: Unlike TF-IDF Vectorization, CountVectorizer does not consider the importance or significance of words in the documents. It treats all words equally in terms of their significance, potentially overlooking the discriminative power of certain terms. This limitation means that common words may receive higher weights, even if they are not particularly informative or distinctive.

Potential Overemphasis on Common Words: CountVectorizer assigns counts to words based solely on their occurrence, which means that commonly occurring words may receive higher weights. While this can be informative in some cases, it may also lead to an overemphasis on common words that do not contribute significantly to distinguishing between documents. This limitation can affect the accuracy and effectiveness of subsequent analysis and modelling tasks.

5.3 HashingVectorizer

In our research, we also employed the HashingVectorizer technique as a memory-efficient vectorization method. Its purpose is to convert text into a fixed-length representation using a hashing function.

5.3.1 Purpose

HashingVectorizer is designed to address the memory usage challenges associated with traditional vectorization techniques. By utilizing a hashing function, it generates a fixed-length representation of the text data, eliminating the need to store a vocabulary dictionary.

5.3.2 Advantages

Memory Efficiency: HashingVectorizer is highly efficient in terms of memory usage. Since it does not require storing a vocabulary dictionary, it can handle large text datasets without significantly impacting memory resources. This makes it particularly suitable for scenarios where memory constraints are a concern.

Scalability: Due to its memory-efficient nature, HashingVectorizer is well-suited for processing large text datasets. It can handle vast amounts of text data without excessive memory consumption, making it a valuable tool for analysing and modelling extensive collections of documents.

5.3.3 Limitations

Hash Collisions: One limitation of HashingVectorizer is the possibility of hash collisions. Hash functions map different words to the same hash value, which can lead to a loss of information. In such cases, distinct words may be incorrectly mapped to the same representation, potentially affecting the accuracy of subsequent analyses.

No Inverse Transformation: Unlike some other vectorization techniques, HashingVectorizer does not provide an inverse transformation to recover the original text from the fixed-length representation. Therefore, it may not be suitable for scenarios where the reconstruction of the original text is necessary.

We selected HashingVectorizer as an alternative vectorization technique in our research to address the challenge of efficiently handling large text datasets. By exploring Hashing Vectorizer's trade-off between memory usage and vectorization performance, we aimed to evaluate its suitability for our specific research objectives.

The memory efficiency of HashingVectorizer allows us to process and analyse extensive collections of text data without overwhelming computational resources. By utilizing this technique, we can assess the impact of memory optimization on the overall performance of subsequent analysis and modelling tasks.

In conclusion, our research project delved into the realm of vectorization techniques for transforming text data into numerical representations. Through our exploration, we utilized TF-IDF Vectorization as the primary technique, CountVectorizer for comparison, and HashingVectorizer to handle large text datasets efficiently.

The selection of these vectorization techniques was guided by their proven effectiveness in text classification tasks and their alignment with our research objectives. By leveraging multiple techniques, we sought to evaluate their respective performances and analyse the impact of different approaches on the classification results.

TF-IDF Vectorization emerged as the focal technique due to its ability to capture word importance and effectively handle stop words. By assigning weights based on term frequency and inverse document frequency, TF-IDF enabled us to highlight significant terms for classification purposes. Furthermore, its consideration of stop words ensured that common words did not overshadow more informative terms.

CountVectorizer served as a valuable point of comparison, offering a straightforward approach to vectorization by counting the occurrence of words. This allowed us to assess the impact of term weighting in TF-IDF compared to a simpler count-based representation. While CountVectorizer lacks importance weighting and may assign higher weights to commonly occurring words, it provided insights into the frequency distribution of terms within the documents.

To address the challenges posed by large text datasets and memory constraints, we employed HashingVectorizer. This memory-efficient technique converted text into a fixed-length representation using a hashing function. By eliminating the need for a vocabulary dictionary, HashingVectorizer enabled us to handle extensive collections of documents without excessive computational resources. However, it is important to note the potential limitation of hash collisions and the lack of an inverse transformation for recovering the original text.

The integration of these vectorization techniques allowed us to comprehensively evaluate their performances and understand the nuances associated with different approaches. Through empirical analysis and comparison, we gained valuable insights into the strengths and limitations of each technique, empowering us to make informed decisions in subsequent modelling and analysis processes.

Overall, the careful selection and utilization of these vectorization techniques contributed to the robustness and effectiveness of our research. By considering their purpose, advantages, limitations, and relevance to our research objectives, we were able to navigate the intricacies of transforming textual data into meaningful numerical representations.

6. Perceptron

Perceptron see [12] and [22], a neural network link incorporating computations and Artificial Intelligence, is utilized for feature tracking input data. It connects to artificial neurons through simple logic gates with binary outputs. Artificial neurons function similarly to biological neurons, with the node representing the cell nucleus, inputs corresponding to dendrites, weights analogous to synapses, and outputs mirroring axons.

In the realm of machine learning, a binary classifier is a model trained to classify data into one of two possible categories. These categories are typically represented as binary labels, such as 0 or 1, true or false, or positive or negative. For instance, a binary classifier can be trained to differentiate between spam and non-spam emails or predict the legitimacy of credit card transactions.

Binary classifiers serve as fundamental components in various machine learning applications see [14]. Multiple algorithms can be employed to construct them, including logistic regression, support vector machines (SVMs), decision trees, random forests, and neural networks. These models are trained using labeled data, where the correct category for each example in the training set is known. Subsequently, they are applied to predict the category of new, unseen examples.

The performance evaluation of a binary classifier encompasses metrics like accuracy, precision, recall, and F1 score. These metrics assess the model's ability to accurately identify positive and negative examples within the data. High-quality binary classifiers are crucial for numerous applications, including natural language processing, computer vision, fraud detection, and medical diagnosis, among others.

6.1 Biological neurons

Biological neurons play a crucial role in the human brain, where they form interconnected networks responsible for processing and transmitting chemical and electrical signals. These neurons consist of various components, including dendrites, which receive information from other neurons, the cell nucleus (or soma), which processes the received information, the axon, which serves as a communication pathway for transmitting information, and synapses, which connect the axon of one neuron to the dendrites of another, see [12] and [22].

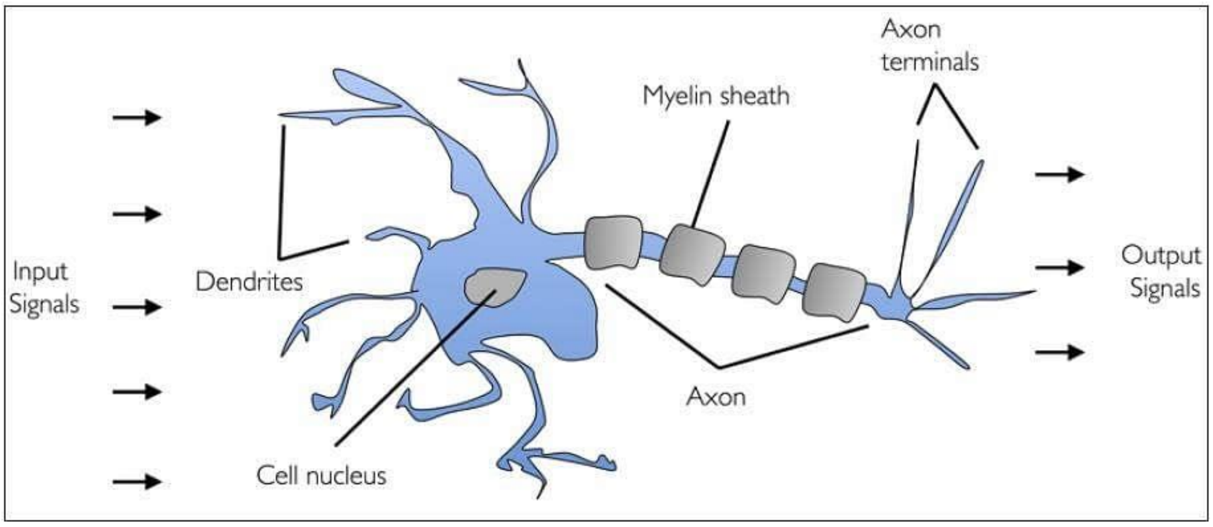


FIGURE 3 BIOLOGICAL NEURON, SOURCE[22]

Inspired by the functioning of biological neurons, researchers Warren McCullock and Walter Pitts introduced the concept of artificial neurons in 1943. They developed the McCullock-Pitts (MCP) neuron, which can be seen as a simplified brain cell functioning as a logic gate with binary outputs.

6.2 Artificial neurons

Artificial neurons are mathematical models based on the behavior of biological neurons. Each artificial neuron receives inputs, assigns weights to these inputs, sums them up, and passes the resulting sum through a nonlinear function to produce an output. This mathematical function mimics the integration and processing of information performed by biological neurons.

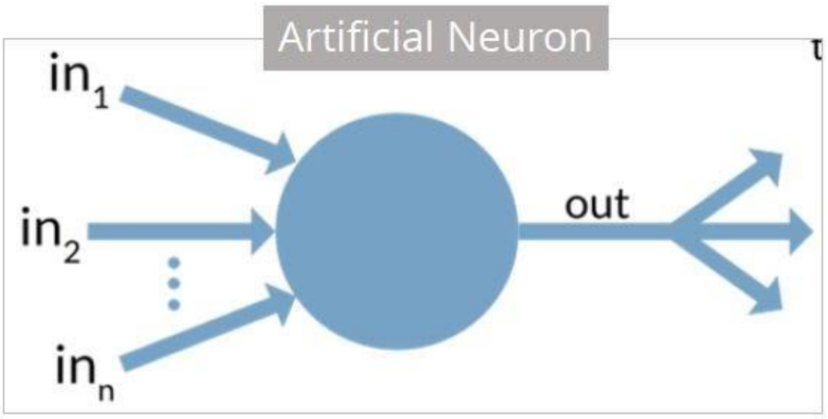


FIGURE 4 ARTIFICIAL NEURON, SOURCE[22]

The development of artificial neurons has opened doors to the creation of artificial neural networks, where multiple artificial neurons are interconnected to perform complex tasks such as pattern recognition, data classification, and decision-making. These artificial neural networks have proven to be powerful tools in various fields, including machine learning, image processing, and artificial intelligence.

In summary, artificial neurons replicate the fundamental principles of biological neurons in a mathematical model, enabling the construction of artificial neural networks that exhibit similar capabilities to process and analyze information.

6.3 Biological Neuron vs. Artificial Neuron

The biological neuron is analogous to artificial neurons in the following terms:

Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

Introduced by Frank Rosenblatt in 1957, the Perceptron algorithm builds upon the original MCP neuron concept. It serves as an algorithm for supervised learning of binary classifiers. The Perceptron learning rule allows neurons to learn and process elements in the training set individually, one at a time. This iterative approach facilitates the training of the Perceptron model, enabling it to make predictions and classify input data into one of two possible categories. The Perceptron algorithm has been widely applied in various fields, including pattern recognition, image classification, and natural language processing, and has proven to be an effective tool in binary classification tasks.

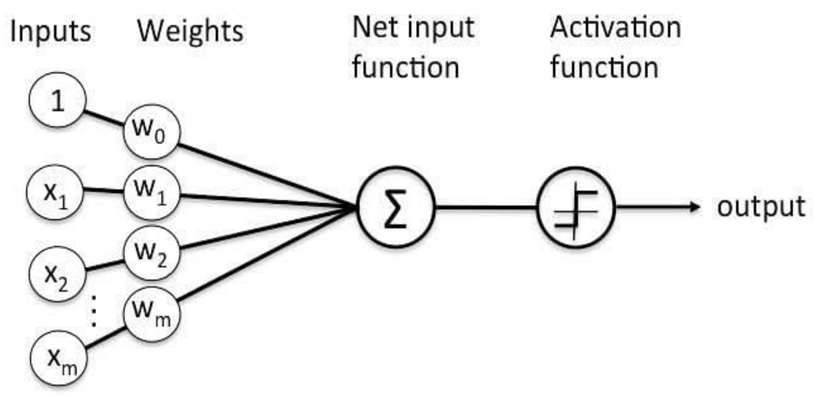


FIGURE 5 PERCEPTRON, SOURCE[22]

The perceptron is a fundamental concept in machine learning and artificial neural networks. It consists of several basic components that enable it to perform binary classification tasks:
Input Layer: The perceptron's input layer consists of one or more input neurons that receive input signals from the external world or other layers of the neural network.

Weights: Each input neuron in the perceptron is associated with a weight, which represents the strength of the connection between the input neuron and the output neuron.

- Bias: A bias term is added to the perceptron's input layer to provide flexibility in modelling complex patterns in the input data.
- Activation Function: The activation function determines the output of the perceptron based on the weighted sum of the inputs and the bias term. Common activation functions used in perceptrons include the step function, sigmoid function, and ReLU function.

- Output: The output of the perceptron is a single binary value, either 0 or 1, indicating the class or category to which the input data belongs.
- Training Algorithm: The perceptron is typically trained using a supervised learning algorithm, such as the perceptron learning algorithm or backpropagation. During training, the weights and biases of the perceptron are adjusted to minimize the error between the predicted output and the true output for a given set of training examples.

The perceptron can be further categorized into single layer perceptrons and multilayer perceptrons. Single layer perceptrons can only learn linearly separable patterns, while multilayer perceptrons can learn about two or more layers, providing greater processing power and the capacity to handle more complex tasks.

The history of the perceptron dates to 1958 when Frank Rosenblatt introduced it as an artificial neural network capable of learning and performing binary classification tasks. Although initial research faced limitations, the development of backpropagation in the 1980s revived interest in artificial neural networks and led to significant advancements in machine learning.

The operation of the perceptron can be understood by examining its underlying mathematical calculations and the application of the activation function. As previously mentioned, the perceptron is considered a single-layer neural link with four main parameters.

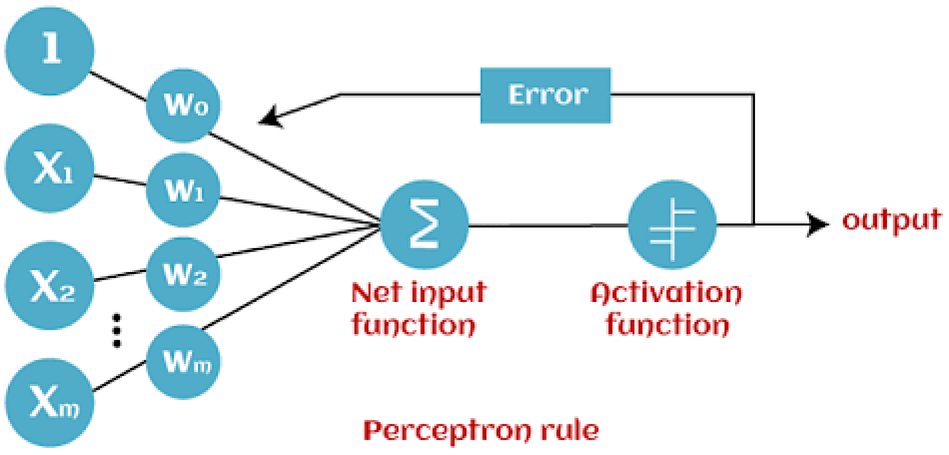


FIGURE 6 PERCEPTRON RULE, SOURCE[22]

Weighted Sum Calculation: The perceptron begins by multiplying each input value with its corresponding weight and then summing these weighted values. Mathematically, this can be expressed as:

$$\sum w_i * x_i = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + \ldots\ldots\ldots x_n * w_n$$, source[22]

Bias Term: To enhance the performance of the perceptron, a bias term 'b' is added to the weighted sum calculated in the previous step:

$$\sum w_i * x_i + b.$$, source[22]

Activation Function: The output of the perceptron is determined by applying an activation function, often referred to as the step function or 'f', to the weighted sum plus the bias term. The activation function maps the result to a desired range, typically (0, 1) or (-1, 1), depending on the specific problem. It is important to note that the weights represent the strength of the connections between the input values and the perceptron, while the input values can shift the activation function curve up or down. The output signal is as follows.

$$Y=f\ (\sum w_i * x_i + b),\ \text{source[22]}$$

where f is an activation function.

6.4 Types of perceptrons

Various types of perceptron models have been developed to address different learning tasks and data complexities, see [12], [22] and [13]. Here are some commonly known types of perceptron models:

Single-Layer Perceptron: The single-layer perceptron, also known as the Rosenblatt perceptron, is the simplest form of perceptron. It consists of a single layer of artificial neurons (perceptrons) that can only learn linearly separable patterns. It is primarily used for binary classification tasks and employs a step function as the activation function.

- The multi-layer perceptron extends the capabilities of the single-layer perceptron by introducing one or more hidden layers between the input and output layers. Each layer contains multiple artificial neurons that perform computations. MLPs can learn complex patterns and can approximate any non-linear function. They utilize various activation functions, such as sigmoid, tanh, or ReLU, to introduce non-linearities into the model.
- A feedforward neural network is a type of perceptron model where the information flows in one direction, from the input layer to the output layer. It can have one or more hidden layers and is capable of learning complex mappings between inputs and outputs. Backpropagation, a popular learning algorithm, is often used to train feedforward neural networks.
- An RBFN is a type of perceptron model that uses radial basis functions as activation functions. It consists of multiple hidden neurons that are centred around specific points in the input space. The RBFN computes the distance between the input and these centres and uses the values as inputs to the activation functions. RBFNs are particularly useful for pattern recognition and function approximation tasks.
- Probabilistic Neural Networks (PNN) are perceptron models that employ a probabilistic approach to classification. They utilize kernel density estimation to estimate the probability density function of each class and make predictions based on the highest probability. PNNs are especially effective when dealing with high-dimensional and noisy data.
- Self-Organizing Map (SOM), also known as a Korhonen network, is a type of unsupervised learning model based on the principles of competitive learning. It consists of an input layer and a competitive layer of neurons arranged in a grid-like structure. SOMs are primarily used for clustering and visualization tasks, allowing the identification of topological relationships in the input data.

These are just a few examples of perceptron models, each tailored for specific learning tasks and data characteristics. The choice of perceptron model depends on the complexity of the problem, the available data, and the desired output. Researchers and practitioners continue to explore and develop new variations and enhancements to perceptron models to tackle diverse machine learning challenges.

6.5 Characteristics of the Perceptron Model

The perceptron model possesses several key characteristics that define its behaviour and capabilities , see [12] and [22]. Understanding these characteristics is crucial for comprehending the functioning and limitations of perceptron models. Here are some important characteristics of the perceptron model:

- Binary Classification: The perceptron model is primarily used for binary classification tasks. It classifies input data into two distinct categories or classes based on the learned weights and activation function. The output of the perceptron is a binary value, typically 0 or 1, representing the predicted class.
- Linear Separability: The perceptron model assumes linear separability of the input data. It can only learn patterns that are linearly separable, where a hyperplane can be used to separate the input data points of different classes. This limitation means that the perceptron may struggle to handle complex nonlinear patterns.
- Supervised Learning: The perceptron model employs supervised learning, where it learns from labelled training data. During the training phase, the model adjusts its weights and biases based on the input-output pairs provided in the training set. It aims to minimize the error between the predicted output and the true output, optimizing its classification performance.
- Activation Function: An activation function is a critical component of the perceptron model. It introduces non-linearity and maps the weighted sum of inputs to the output of the perceptron. Commonly used activation functions in perceptrons include the step function (for binary output), sigmoid function, and rectified linear unit (ReLU) function. The choice of activation function affects the model's ability to capture complex patterns.
- Threshold Decision Rule: The perceptron model employs a threshold decision rule to determine the predicted class. If the weighted sum of inputs exceeds a predefined threshold, the perceptron outputs one class; otherwise, it outputs the other class. This decision rule allows the perceptron to make binary decisions based on the activation function's output.
- Online Learning: The perceptron model supports online learning, where it updates its weights and biases after processing each training example individually. It iteratively learns from one training example at a time, making it suitable for scenarios with large datasets or streaming data. This characteristic enables efficient and incremental learning.
- Model Interpretability: Perceptron models are often considered interpretable due to their simplicity and transparency. The learned weights and biases provide insights into the model's decision-making process, allowing analysts to understand the importance of different features in the classification task. This interpretability is advantageous in domains where model transparency is essential.
- Model Limitations: The perceptron model has certain limitations. It may struggle with complex classification tasks that involve non-linear patterns, as it assumes linear separability. Additionally, the model may suffer from the "perceptron convergence theorem" issue, where it fails to converge and find an optimal solution if the classes are not linearly separable.

These characteristics define the perceptron model and shape its behaviour in performing binary classification tasks. While perceptrons have certain limitations, they paved the way for more advanced neural network architectures that can handle complex patterns and non-linear relationships.

6.6 Perceptron Learning Rule

The perceptron learning rule is a fundamental algorithm used to train perceptron models in machine learning. Developed by Frank Rosenblatt in 1957, it serves as a building block for more complex neural network architectures. The learning rule is designed to adjust the weights of the input signals in order to improve the accuracy of the perceptron's predictions , see [12] and [22].

The process of training a perceptron involves iteratively presenting training examples to the model and updating the weights based on the prediction errors. The learning rule follows a gradient descent approach, aiming to minimize the error between the perceptron's predicted output and the desired output for each training example.

Initially, the weights of the perceptron are randomly assigned. As each training example is presented, the perceptron computes a weighted sum of the input signals and applies an activation function to generate the output. The predicted output is then compared to the true output, and the error is calculated.

The learning rule uses the error to adjust the weights. If the perceptron makes a correct prediction, the weights remain unchanged. However, if an error occurs, the weights are adjusted to reduce the error in subsequent iterations. The adjustment is proportional to the learning rate, which determines the step size of weight updates.

By repeating this process for multiple training examples, the perceptron gradually learns to classify inputs correctly. The learning rule updates the weights until the perceptron reaches a point where it achieves satisfactory accuracy on the training data.

It is important to note that the perceptron learning rule works effectively only when the data is linearly separable. If the data is not linearly separable, the perceptron may not converge to a solution. In such cases, more advanced techniques, such as multi-layer perceptrons or support vector machines, are often employed.

In summary, the perceptron learning rule is a powerful algorithm that enables perceptrons to learn from labelled training data. By iteratively adjusting the weights based on prediction errors, the perceptron can effectively classify inputs into different categories. While the perceptron learning rule has its limitations, it serves as a fundamental concept in machine learning and has paved the way for more sophisticated neural network architectures.

6.7 Perceptron Functions

The perceptron model is a fundamental building block in the field of machine learning, particularly in the design and implementation of neural networks. Its operation is based on a few key components and functions, which we will explore in more detail in this section , see [12] and [22].

- **Weighted Input Signals**
  The first step in the operation of a perceptron model involves the input signals. These signals, which represent the data that the perceptron is processing, are each multiplied by a corresponding weight. These weights are crucial as they allow the perceptron to assign different levels of importance to each input signal based on its relevance to the task at hand. The weighted inputs are then aggregated to calculate a weighted sum. This sum forms the basis for the next step in the perceptron's operation.
- **Activation Function**
  The activation function is a critical component of the perceptron model. It introduces non-linearity into the model, which is essential for enabling the perceptron to handle complex patterns and relationships in the data. The activation function takes the weighted sum as input and transforms it into an output that is used for prediction or classification tasks.

6.8 Perceptron Activation Functions

There are several commonly used activation functions in perceptrons, each with its own unique characteristics and uses, see [12] and [22].

- **Step Function**

The step function is a binary activation function. It produces one of two possible outputs based on whether the input (the weighted sum) is above or below a certain threshold. This makes the step function particularly useful for binary classification tasks, where the goal is to categorize data into one of two classes or categories.

- **Sigmoid Function**
  The sigmoid function is another popular activation function. It maps the weighted sum to a value between 0 and 1, providing a probability-like output. This output can be interpreted as the likelihood of a certain class or category, making the sigmoid function especially useful for tasks that involve estimating probabilities or dealing with binary classification problems.
- **Rectifier Function (ReLU)**
  The Rectified Linear Unit (ReLU) function is a commonly used activation function in deep learning models. It outputs the input directly if it is positive; otherwise, it outputs zero. This function introduces non-linearity without requiring expensive computational operations.
- **Hyperbolic Functions**
  Such as the hyperbolic tangent (tanh), can also be used as activation functions. Like the sigmoid function, they map inputs to a value between -1 and one, but they offer the advantage of being zero-centred, which can help speed up learning.

By incorporating an activation function, the perceptron model becomes capable of capturing non-linear relationships between inputs and outputs. This non-linearity is crucial for the perceptron's ability to learn complex patterns and make accurate predictions, making it a versatile tool for a wide range of machine learning tasks.6.9 Output of Perceptron
The output of a perceptron is determined by the activation function, which transforms the weighted sum of inputs into a meaningful output. In binary classification tasks, the output of the perceptron is typically represented as either 0 or 1, indicating the predicted class or category of the input data. The output value represents the perceptron's decision or prediction based on the learned weights and biases. By comparing the output to a threshold or using a probabilistic interpretation, the perceptron can classify new examples and make predictions based on the learned patterns in the training data.

6.9 Perceptron Error

Understanding the concept of error in the context of a perceptron model is crucial for comprehending how these models learn and adapt to improve their performance over time. The error in a perceptron is a measure of the difference between the model's predicted output and the true output for a given training example. This section will delve deeper into the nature of perceptron error, its implications, and how it is used to enhance the model's performance, see [12] and [22].

- **Defining Perceptron Error**
  At its core, perceptron error quantifies the discrepancy between the perceptron's classification and the desired classification. In other words, it measures how far off the perceptron's prediction is from the actual truth. This error can be calculated in various ways, depending on the specific implementation of the perceptron model. However, a common method is to subtract the predicted output from the true output, providing a simple yet effective measure of the model's performance on a specific training example.
- **Implications of Perceptron Error**
  The magnitude and direction of the perceptron error have significant implications. A large error indicates that the perceptron's current weights and biases are not effective at

classifying the given input correctly. On the other hand, a small error suggests that the perceptron is performing well on the specific training example. The direction of the error (positive or negative) can provide insights into how the weights and biases need to be adjusted to improve the model's performance.

- **Minimizing Perceptron Error**
  The primary goal of the perceptron learning rule is to minimize the perceptron error. This is achieved by iteratively adjusting the weights and biases of the perceptron during the training process. If the perceptron's prediction is correct, the weights and biases remain unchanged. However, if there is an error, the weights and biases are adjusted in a way that makes the perceptron's output closer to the true output. This adjustment is typically proportional to the magnitude of the error, meaning larger errors result in larger adjustments.

The process of minimizing the perceptron error is essentially a process of optimization. The perceptron learning rule uses a method called gradient descent, which iteratively adjusts the weights and biases to move towards the minimum of the error function. This iterative process continues until the perceptron error is minimized to a satisfactory level or until a predetermined number of iterations have been completed.

In summary, the concept of perceptron error is central to the operation and learning process of a perceptron model. By quantifying the discrepancy between the perceptron's predictions and the true outputs, and by iteratively adjusting the model's parameters to minimize this error, the perceptron can improve its performance and make increasingly accurate predictions.

6.10 Perceptron Implementing Logic Gates

Perceptrons are powerful tools that can be used to implement basic logic gates, enabling them to perform logical operations such as AND, OR, and NOT. The ability to implement logic gates with perceptrons highlights their fundamental role in computational tasks and showcases their capacity for decision-making, see [12] and [22].

To implement logic gates using perceptrons, the weights and bias of the perceptron need to be appropriately set. The weights determine the importance or significance of each input in deciding, while the bias allows for fine-tuning and adjusting the decision boundary. By carefully configuring these parameters, the perceptron can learn the appropriate decision boundary that separates the input space into the desired classes, effectively emulating the behaviour of the logic gate. For example, in the case of an AND gate, the perceptron needs to assign higher weights to the inputs that correspond to TRUE values and a bias that enables the perceptron to produce a TRUE output only when all inputs are TRUE. Similarly, for an OR gate, the perceptron assigns non-zero weights to the inputs and sets a bias to produce a TRUE output if any of the inputs are TRUE.

The perceptron's ability to implement logic gates is not limited to simple binary operations. By combining multiple perceptrons or using more complex architectures, perceptrons can emulate more sophisticated logical operations, such as XOR (exclusive OR) and NAND (NOT AND) gates. This demonstrates the versatility and flexibility of perceptrons in performing a wide range of computational tasks.

Implementing logic gates with perceptrons not only showcases their practical utility but also provides insights into their underlying computational principles. By training perceptrons to mimic logical behaviour, we can better understand their decision-making processes and the ways in which they process and transform input information.

Overall, the ability of perceptrons to implement basic logic gates underscores their significance in the field of artificial intelligence and machine learning. It highlights their capacity to perform logical operations and lays the foundation for more complex neural network architectures used in modern deep learning systems.

Perceptrons are fundamental units in neural networks used for binary classification tasks. They learn from labelled training data and make predictions based on weighted inputs and an activation function. Perceptrons can implement basic logic gates and have advantages in terms of interpretability and transparency. The choice of activation function, such as sigmoid, rectifier, or hyperbolic functions, impacts the behaviour and performance of perceptrons.

6.11 The future of perceptrons

holds immense potential for advancements in the field of artificial intelligence and machine learning. While perceptrons have already played a significant role in laying the foundation for neural network architectures, there are several avenues for further exploration and enhancement. , see [12] and [22].

One area of focus for the future of perceptrons lies in the development of more sophisticated learning algorithms. While the perceptron learning rule has been effective for linearly separable data, it may face challenges in handling complex non-linear patterns. Researchers are actively exploring novel algorithms that can enable perceptrons to learn from and classify data that is not easily separable by a linear decision boundary. These advanced learning algorithms may leverage techniques such as deep learning, reinforcement learning, or evolutionary algorithms to enhance the perceptron's ability to handle more complex tasks.

Another aspect of perceptron advancement involves the exploration and incorporation of advanced activation functions. While traditional activation functions such as the step function, sigmoid function, and rectified linear unit (ReLU) have been widely used, there is ongoing research to discover and develop new activation functions that can improve the performance and capabilities of perceptrons. These new functions may introduce additional non-linearities or adaptively adjust their shape to capture intricate patterns in the data, enabling perceptrons to better handle complex relationships and improve their accuracy.

Furthermore, addressing the limitations of perceptrons in handling complex non-linear patterns is a crucial direction for future research. One approach involves the utilization of multi-layer perceptrons (MLPs) or deep neural networks, which extend the capabilities of single-layer perceptrons. MLPs employ multiple hidden layers of perceptrons, allowing them to learn more complex representations of the input data. Deep learning architectures, consisting of multiple layers with non-linear activation functions, enable perceptrons to capture hierarchical and abstract features, making them capable of handling intricate and high-dimensional data.

Additionally, the future of perceptrons involves leveraging advancements in hardware and computational capabilities. With the increasing availability of specialized hardware, such as graphics processing units (GPUs) and tensor processing units (TPUs), perceptron models can be trained and deployed more efficiently, enabling faster and more scalable processing of complex tasks. This opens up possibilities for real-time applications and the utilization of perceptrons in resource-constrained environments.

Moreover, the integration of perceptrons with other emerging technologies can lead to exciting advancements. For instance, combining perceptrons with natural language processing techniques can enhance language understanding and enable more advanced conversational agents. Integrating perceptrons with computer vision systems can lead to improved object recognition and scene understanding. By leveraging the synergy between perceptrons and other AI technologies, we can unlock new opportunities for intelligent systems across various domains.

In summary, the future of perceptrons encompasses a wide range of research directions and advancements. From developing more sophisticated learning algorithms and activation functions to addressing the limitations of linear separability and harnessing the power of deep learning architectures, perceptrons will continue to evolve and contribute to the advancement of artificial intelligence and machine learning. By embracing these future prospects, researchers and practitioners can unlock new frontiers in pattern recognition, data analysis, and decision-making, leading to innovative applications and transformative breakthroughs.

7. Artificial intelligence tool for detecting fake news

These days, machine learning is one of the fastest-growing areas in computer science and has found applications in different businesses, from healthcare and back to retail and fabricating. Machine learning calculations have the capacity to examine endless sums of information and extricate important experiences, which can help businesses make superior choices and move forward their items and administrations, see[28] and [45].

In later a long time, machine learning has too developed as a capable device for distinguishing and classifying abhor discourse and hostile dialect on social media stages. As increasingly individuals utilize social media to communicate and share their sees, the issue of despise discourse has ended up progressively predominant. Businesses are especially concerned approximately the effect of despise discourse on their notoriety and brand picture, and numerous are turning to machine learning as an arrangement.

Machine learning calculations can examine huge volumes of printed information and learn to recognize between despise discourse and non-hate discourse based on designs and highlights within the information. These calculations can to adjust and make strides over time as they are uncovered to more information, making them an important instrument for handling the issue of despise discourse on social media.

This chapter will investigate the instruments of machine learning for despise discourse and hostile dialect classification on Twitter and normal dialect preparing. It will give an outline of the different machine learning calculations utilized for this reason and talk about their qualities and restrictions.

7.1 Artificial intelligence and machine learning basics

Machine learning (ML) has revolutionized decision-making and pattern recognition by automating tasks that previously required human intellect. According to [9], [13] and [45] ML can be categorized into three main types:

1. **Supervised learning:** In supervised learning, the machine is trained on a dataset where each data point is labelled with a target or outcome variable. By learning from this labelled data, the machine can make predictions or decisions. Examples of supervised learning algorithms include k-nearest neighbours, classification trees, and certain types of neural networks.
2. **Unsupervised learning**: Unsupervised learning involves training the machine on an unlabelled dataset, where there are no predefined target variables. The machine autonomously discovers patterns or structures within the data. Clustering algorithms like k-means, dimensionality reduction techniques, and some neural networks fall under the category of unsupervised learning.
3. **Reinforcement learning:** In reinforcement learning, the machine learns through interaction with its environment and receives feedback in the form of rewards or penalties. It aims to maximize long-term rewards by taking actions that lead to positive outcomes. Reinforcement learning finds applications in game-playing AI agents and autonomous robots.

In addition to these three categories, there is another type called semi-supervised learning, which combines aspects of supervised and unsupervised learning. For instance, in the detection of hate speech, an ML model can be trained using both labelled and unlabelled examples.

However, for this work, supervised learning algorithms are utilized as the class labels for the tweets are already known.
These various ML techniques enable machines to learn, recognize patterns, and make informed decisions, contributing to advancements in a wide range of fields. By leveraging supervised, unsupervised, and reinforcement learning, researchers and practitioners can develop sophisticated models to tackle complex problems and enhance decision-making processes.

7.2 Natural language processing

Natural language processing (NLP) is an interdisciplinary field that combines computer science and artificial intelligence to enable computers to understand, interpret, and generate human language. Its purpose is to develop models and algorithms that can process and analyse natural language data, including text, speech, images, and even video or audio. NLP incorporates various techniques such as machine learning, deep learning, and linguistic analysis to achieve its goals, see[24], [25] and [46].
The applications of NLP are extensive and diverse, including language translation, sentiment analysis, chatbots, speech recognition, and text summarization, among others. The ultimate objective of NLP is to create machines that possess a deep understanding of human language and can interact with humans in a natural and meaningful manner.
However, NLP poses significant challenges due to the complexity of human language, particularly its syntax and latent elements such as irony, humour, sarcasm, and expressions. The semantic meaning of words can vary significantly depending on these contextual factors, making it difficult to define precise rules to capture their nuances. NLP techniques need to address these challenges and devise strategies to overcome them effectively. The typical NLP pipeline involves multiple phases. The initial step is to convert raw text input into numerical representations called vectors. This process, known as tokenization, involves dividing the text into smaller units called tokens, which are typically a few words or symbols. Additionally, pre-processing steps like stop-word removal, normalization of words by lowercase conversion and punctuation removal, are commonly applied. The resulting numerical vectors are then used as input for machine learning classifiers to train and evaluate models.

7.3 Classifiers

All the classifiers that have been used in this thesis are based on Neural networks, which is made from: Perceptron: In the field of artificial neural networks, a perceptron refers to a computational unit that emulates the behaviour of a biological neuron. It takes multiple input signals, each associated with a weight, and combines them linearly using summation. The perceptron then applies an activation function to the weighted sum to produce an output signal. This activation function introduces non-linearity, allowing the perceptron to model complex relationships in the data, see[25] and [24].
Input Signals: In a neural network, input signals represent the features or attributes of the input data. These signals can be numerical values, pixel intensities, or any other form of data representation. Each input signal is associated with a weight, which determines its relative importance in the network's computations. The weights are adjusted during the learning process to optimize the network's performance.
Output Signal: The output signal of a perceptron or neuron represents the computed result or prediction made by that neuron. It is obtained by applying the activation function to the weighted sum of the input signals. The output signal is a transformed representation of the input

data and carries information about the learned patterns or predictions made by the neural network.
Layers: Neural networks are organized into layers, which are composed of interconnected neurons. The main layers in a neural network are the input layer, hidden layers, and output layer. The input layer receives the initial input signals, while the hidden layers perform intermediate computations by applying weights, activation functions, and biases to the input signals. The output layer produces the final output signals or predictions.
Weights: Weights are parameters associated with the connections between neurons in a neural network. They determine the strength and direction of the influence that each input signal has on the neuron's output. During the training process, the weights are adjusted iteratively using optimization algorithms such as backpropagation. This adjustment allows the network to learn from the data and improve its ability to make accurate predictions.
Activation Function: An activation function is a mathematical function that introduces non-linearity to the computations of a neuron. It is applied to the weighted sum of the input signals and determines the neuron's output or activation level. Common activation functions include the sigmoid function, which maps the input to a value between 0 and 1, and the rectified linear unit (ReLU), which sets negative inputs to zero and keeps positive inputs unchanged. Activation functions enable neural networks to model complex relationships and capture non-linear patterns in the data.
Bias: Bias is an additional parameter in a neuron that represents a constant input value. It allows the neuron to adjust its output independently of the input signals. Bias helps the network to learn and make predictions even when all input values are zero. By introducing a bias term, the neuron can shift the activation function's curve, enabling it to model data that is not cantered around zero.
Feedforward: Feedforward refers to the process of propagating input signals through the neural network in a forward direction, from the input layer to the output layer. During feedforward, each neuron receives input signals, applies weights and biases, computes the weighted sum, and passes the result through the activation function. This process is repeated for all neurons in each layer, allowing the network to transform the input data and produce the final output.
Backpropagation: Backpropagation is a learning algorithm used to train neural networks. It involves propagating the error or difference between the network's predicted output and the desired output backward through the network. The error is used to adjust the weights and biases of the neurons iteratively, following the gradient of the error function. By iteratively updating the weights and biases, the network can minimize the difference between its predictions.
In our research, we utilized a diverse set of classifiers and models to tackle the task of identifying fake news. The selected approaches included PassiveAggressiveClassifier, Logistic Regression, SGDClassifier, Convolutional Neural Network (CNN), and Deep Neural Network (DNN). Each of these classifiers/models has its underlying principles and advantages, which we will explore in detail.

7.1 Passive Aggressive Classifier

The Passive Aggressive Classifier is an online learning algorithm specifically designed for binary classification tasks. It draws inspiration from fundamental concepts in neural networks, such as the perceptron model, weights, activation functions, and layers. see[25] and [24] and [23].

The perceptron model forms the basis of the Passive Aggressive Classifier's learning mechanism. In this model, each input feature is associated with a weight that represents its importance in the classification process. These weights are iteratively updated during the learning process based on observed errors. The activation function, typically a threshold function, is applied to the weighted sum of the input features to determine the classification decision.

The Passive Aggressive Classifier extends this concept by introducing a unique update strategy. When the classifier makes a correct prediction, it updates its model parameters passively, preserving the existing weights. However, when a misclassification occurs, the classifier updates its parameters aggressively to correct the error. This adaptive update mechanism allows the classifier to handle evolving data streams and adapt to changing patterns.

The reason for using the Passive Aggressive Classifier is its ability to handle evolving data streams and process large volumes of data efficiently. In scenarios where new observations continuously arrive, incremental learning becomes crucial. The online learning nature of the Passive Aggressive Classifier enables it to update its model based on new observations, making it suitable for real-time applications.

To illustrate the effectiveness of the Passive Aggressive Classifier, let us consider an example involving a dataset of news articles labelled as either real or fake. The classifier's goal is to learn from this data and make accurate predictions on new, unseen articles. By employing the Passive Aggressive Classifier, we can iteratively update the model's weights and thresholds based on observed errors. This adaptive learning process allows the classifier to continuously refine its decision boundaries and improve its classification accuracy over time.

Additionally, the Passive Aggressive Classifier calculates the classification margin, which represents the confidence of its decision. By analysing the magnitude of the margin, we can assess the confidence level of the classifier's predictions and evaluate its performance on different instances.

In summary, the Passive Aggressive Classifier combines the principles of the perceptron model and online learning to adaptively update its model parameters based on observed errors. This makes it a powerful tool for handling evolving data streams and incremental learning scenarios. By incorporating the Passive Aggressive Classifier into our research, we aimed to leverage its adaptive nature and explore its effectiveness in identifying fake news.

7.2 Logistic Regression

Logistic Regression is a fundamental linear classifier commonly used for binary classification tasks. It leverages the logistic function, also known as the sigmoid function, to model the probability of the binary outcome. Logistic Regression can be seen as a simplified neural network with one input layer, one output layer, and a non-linear activation function. see[25], [24] and [23].

In logistic regression, each input feature is associated with a weight, and the model applies a logistic/sigmoid function to the weighted sum of the input features to compute the predicted probability. The logistic function maps the linear combination of input features to a value between 0 and 1, representing the probability of belonging to the positive class.

Logistic Regression also applies regularization techniques, such as L1 or L2 regularization, to prevent overfitting and improve generalization. These regularization terms penalize large coefficients, encouraging the model to favour simpler explanations and reduce the risk of capturing noise in the data.

One advantage of Logistic Regression is its interpretability. We can examine the coefficients associated with each input feature to gain insights into their influence on the classification outcome.
In our research, Logistic Regression serves as a benchmark model against which we compare the performance of other classifiers and models. By establishing an interpretable baseline, we can assess the effectiveness and complexity of alternative approaches. Furthermore, Logistic Regression allows us to examine linear relationships between the input features and the target variable, providing valuable insights into the significance of different factors.
To illustrate the application of Logistic Regression, let us consider an example related to fake news detection. Suppose we have a dataset containing news articles labelled as either real or fake, along with various features such as the article's word count, readability score, and the presence of certain keywords. By training a logistic regression model on this dataset, we can estimate the probabilities of an article being classified as real or fake based on the provided features. Additionally, we can interpret the coefficients of the logistic regression model to understand the impact of each feature on the classification decision.
In summary, Logistic Regression is a widely used linear classifier that leverages the logistic function to model the probability of a binary outcome. Its interpretability, ability to capture linear relationships, and regularization techniques make it a valuable tool in binary classification tasks. By employing Logistic Regression as a benchmark model, we can compare the performance of alternative classifiers and models and gain insights into the influence of different factors on the classification outcome.

7.3 SGDClassifier

The SGDClassifier, short for Stochastic Gradient Descent Classifier, is an efficient linear classifier commonly used in machine learning for binary and multiclass classification tasks. It is trained using the stochastic gradient descent optimization algorithm, which updates the model parameters iteratively based on a fraction of the training data at each step, see[23] and [24] and [25].The SGDClassifier's ability to process data in small batches and update the model parameters incrementally contributes to its computational efficiency. This efficiency is particularly valuable in text classification tasks that often involve vast amounts of data, such as analysing large corpora of text documents or processing real-time streams of textual information.
One advantage of the SGDClassifier is its flexibility in handling various loss functions and regularization techniques. We can choose different loss functions, such as hinge loss for linear SVM or log loss for logistic regression, depending on the specific classification task. Additionally, the SGDClassifier supports different types of regularization, including L1 and L2 regularization, which help prevent overfitting and enhance generalization.
In our research, we employ the SGDClassifier to address the problem of identifying fake news. By utilizing its computational efficiency, we can efficiently process large text datasets and handle the demands of real-time analysis. Furthermore, the SGDClassifier's ability to adapt to changing data and update model parameters incrementally aligns well with scenarios where computational resources are limited or where the classification system needs to continuously learn from new observations.
To illustrate the application of the SGDClassifier, let us consider an example related to sentiment analysis of customer reviews. Suppose we have a dataset containing customer reviews labelled as positive or negative, along with the corresponding textual content. By

training an SGDClassifier on this dataset, we can learn a linear decision boundary that separates positive and negative reviews. The stochastic gradient descent algorithm updates the model parameters by iteratively considering a subset of the training data, allowing us to efficiently learn the classification boundary even with large-scale datasets.

In summary, the SGDClassifier is an efficient linear classifier trained using stochastic gradient descent. Its ability to process data in small batches, handle large-scale datasets, and adapt to changing data makes it a valuable tool in text classification tasks. By utilizing the SGDClassifier in our research, we aim to achieve high classification performance while effectively managing computational complexity, particularly in scenarios where computational resources are limited or real-time processing of large text datasets is required.

7.4 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are powerful deep learning models specifically designed to automatically learn hierarchical patterns and features from input data see[15]. While initially developed for image analysis tasks, CNNs have proven to be highly effective in various domains, including natural language processing and text analysis. By leveraging their unique architecture, CNNs excel at capturing local patterns and spatial dependencies in data, making them particularly suitable for text classification tasks.

CNNs consist of multiple layers, including convolutional layers and pooling layers. In the context of text classification, the input data is typically represented as a sequence of words or characters. The convolutional layers employ filters of varying sizes to convolve over the input text, extracting local patterns and features. These filters act as feature detectors, enabling the network to learn meaningful representations of the input text at different scales. Through the use of non-linear activation functions, such as ReLU (Rectified Linear Unit), CNNs can capture complex relationships and non-linearities in the data.

Pooling layers are employed to perform dimensionality reduction and down sample the learned features. Max pooling, for instance, selects the maximum value within a sliding window, effectively reducing the spatial dimension of the features while retaining the most salient information. This hierarchical structure allows CNNs to capture different levels of abstraction, enabling the model to identify both fine-grained details and high-level textual features that contribute to distinguishing between real and fake news.

In the context of fake news detection, CNNs can effectively learn relevant features from text representations, such as word embeddings or character sequences. By training on a large corpus of labelled data, the CNN can automatically learn to detect patterns indicative of fake news. These patterns can include linguistic cues, stylistic inconsistencies, or even subtle biases in the language used.

By employing CNNs in our research, we aim to leverage their ability to automatically learn relevant features and improve the classification performance of fake news detection. The hierarchical nature of CNNs allows them to capture both local and global patterns, enabling the model to make informed predictions based on a combination of fine-grained details and higher-level semantic information. Moreover, CNNs have demonstrated remarkable success in various natural language processing tasks, indicating their potential for enhancing the accuracy and robustness of fake news detection systems.

In summary, Convolutional Neural Networks (CNNs) are deep learning models specifically designed to learn hierarchical patterns and features from input data. In the context of text classification, CNNs excel at capturing local patterns and spatial dependencies, making them ideal for fake news detection tasks. By employing CNNs, we aim to leverage their ability to automatically learn relevant features from text representations and improve the classification performance of our fake news detection system. The hierarchical structure of CNNs allows them to capture different levels of abstraction, enabling the model to identify both fine-grained and high-level textual features that contribute to distinguishing between real and fake news.

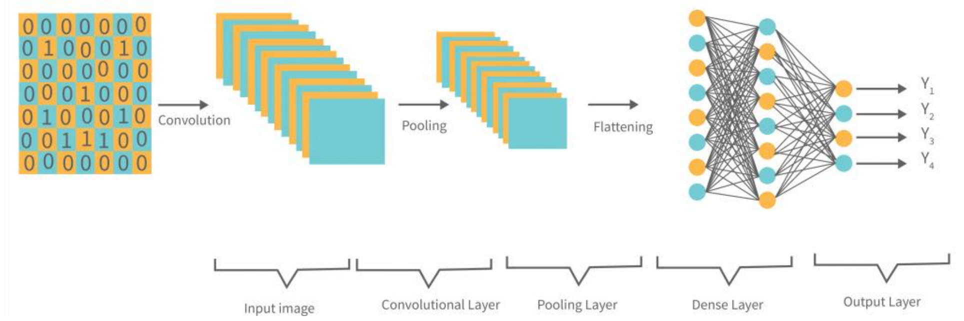


FIGURE 7 TYPICAL CNN ARCHITECTURE, SOURCE [5]

7.5 Deep Neural Network (DNN)

Deep Neural Networks (DNNs) are state-of-the-art machine learning models composed of multiple layers, including multiple hidden layers between the input and output layers, see[15]. DNNs are renowned for their ability to learn complex representations and capture intricate patterns in various domains, including natural language processing and text analysis. By employing DNNs, we aim to leverage their capacity to identify nuanced patterns and enhance the detection of fake news.

In contrast to traditional shallow models, such as logistic regression or linear classifiers, DNNs are capable of capturing intricate relationships and modelling highly nonlinear mappings between input features and target outcomes. This is achieved through the use of nonlinear activation functions, such as the rectified linear unit (ReLU), which introduce nonlinearity into the network's computations. This enables DNNs to learn and represent complex patterns that may exist in the textual data. The multi-layer architecture of DNNs allows for the extraction of high-level abstract features from the input text. Each hidden layer learns to represent increasingly abstract features based on the patterns present in the data. As information flows through the network, lower layers capture simple and local patterns, while higher layers capture more complex and global patterns. This hierarchical feature learning enables the model to capture intricate relationships and identify subtle patterns that may be indicative of fake news.

For instance, a DNN can learn to recognize the co-occurrence of specific words or phrases that are frequently associated with misinformation. By assigning higher weights to these learned features, the model can effectively distinguish between real and fake news articles. Furthermore, the network can learn to identify more subtle linguistic cues, such as inconsistencies in writing style, biased language, or logical fallacies, which are often characteristic of fake news.

Training a DNN involves a process called backpropagation, where the network adjusts its internal parameters, known as weights, based on the error between its predictions and the ground truth labels. This iterative process allows the model to refine its internal representations and improve its classification performance over time. With sufficient training data and computational resources, DNNs have demonstrated remarkable success in various natural language processing tasks, making them a compelling choice for enhancing fake news detection systems.

By incorporating DNNs into our research, we aim to explore their potential in identifying intricate patterns and improving the accuracy of fake news detection. The multi-layer architecture of DNNs enables them to capture high-level abstract features from textual data, facilitating the identification of complex relationships and subtle indicators of fake news. Leveraging their capacity to learn and represent nonlinear mappings, DNNs have the capability to enhance the performance and robustness of fake news detection systems.

In summary, Deep Neural Networks (DNNs) are powerful machine learning models capable of capturing intricate patterns and learning complex representations. By employing DNNs, we aim to leverage their potential in identifying nuanced patterns related to fake news detection. The multi-layer architecture of DNNs allows for the extraction of high-level abstract features from the textual data, enabling the model to capture intricate relationships and identify subtle patterns that may be indicative of fake news. Through the use of nonlinear activation functions and the iterative training process, DNNs can refine their internal representations and improve their classification performance. By exploring the capabilities of DNNs, we seek to enhance the accuracy and effectiveness of fake news detection systems.

The selection of these classifiers and models was based on their individual strengths and suitability for addressing the problem of identifying fake news. The traditional classifiers (PassiveAggressiveClassifier, Logistic Regression, SGDClassifier) were chosen for their interpretability, computational efficiency, and benchmark performance in binary classification tasks. On the other hand, the deep learning models (CNN, DNN) were selected to leverage their ability to capture complex relationships and learn meaningful representations from text data. By employing a combination of classifiers and models, we aimed to leverage their respective strengths and explore their performance in accurately identifying fake news. Additionally, the inclusion of deep learning models allowed us to assess the potential of artificial intelligence techniques in improving the classification results by capturing intricate patterns and semantic relationships in the data.

Regarding the relationship between vectorizers and classifiers, vectorizers play a crucial role in preparing textual data for classification tasks. They are responsible for converting text documents into numerical representations that classifiers can understand and process. Vectorizers, such as CountVectorizer or TF-IDF Vectorizer, transform text data into feature vectors, where each vector represents a document and its respective features (words, n-grams, etc.). These feature vectors serve as input to the classifiers, allowing them to learn patterns and make predictions based on the numerical representations of the text.

In our research, we used vectorization techniques like CountVectorizer and TF-IDF Vectorizer to convert text documents into numerical representations. These vectorizers capture the frequency or importance of words in the documents, respectively.

The resulting feature vectors were then fed into the classifiers (e.g., PassiveAggressiveClassifier, Logistic Regression, SGDClassifier, CNN, DNN) for training and prediction. By combining vectorization and classification techniques, we aimed to create effective models for identifying fake news.

Overall, the vectorizer-classifier relationship is essential in text classification tasks. Vectorizers transform textual data into numerical representations, enabling classifiers to learn from and make predictions on the transformed data. The choice of vectorization technique and classifier can significantly impact the performance and effectiveness of the fake news detection system.

8. Experimental Setup and Evaluation

In our research, we meticulously designed the experimental setup to comprehensively evaluate the performance of various classifiers, models, and vectorizers in identifying fake news across multiple datasets related to war and news see [23]. The experimental choices were made based on established best practices and the specific requirements of the research objectives.

8.1 Train-Test Split

One crucial aspect of the experimental setup was the train-test split, which aimed to ensure a robust evaluation of the models see [2]. We partitioned each dataset into training and testing sets using a standard 80-20 train-test split. This means that 80% of the data was allocated for training the models, while the remaining 20% was kept for evaluating their performance on unseen data. The 80-20 train-test split strikes a balance between providing a sufficiently large training set to effectively train the models and reserving an independent testing set to assess their generalization performance. By using unseen data for evaluation, we can gauge how well the models perform on new and unseen instances, which is crucial for assessing their real-world applicability.

To provide further insights into the evaluation process, we calculated various evaluation metrics, including training error, test error, and generalization error. Training error, also known as training accuracy, measures the performance of the models on the training data itself. It indicates how well the models fit the training data and can be used as a measure of their capacity to capture patterns and features within the dataset.

Test error, on the other hand, assesses the models' performance on the independent testing set. It reflects how well the models generalize to unseen instances and is an essential metric for evaluating their ability to identify fake news in real-world scenarios. By comparing the training and test errors, we can assess the models' tendencies for overfitting or underfitting. Ideally, we aim for models that perform well on both training and testing sets, indicating a good balance between fitting the training data and generalizing to new instances.

Generalization error represents the difference between the training and test errors. It quantifies how much the models' performance degrades when faced with new and unseen instances. A low generalization error suggests that the models can effectively identify fake news across diverse datasets, including those related to war, news, and cards, indicating their potential for real-world applicability.

By carefully implementing the train-test split and evaluating the models using training error, test error, and generalization error, we were able to gain comprehensive insights into their performance. This evaluation methodology allowed us to measure the models' ability to identify fake news across various datasets and assess their generalization capabilities accurately.

8.2 Evaluation Metrics

In addition to the train-test split and the calculation of training, test, and generalization errors, we employed a range of evaluation metrics to assess the performance of the classifiers and models see [6] . These metrics provided a comprehensive understanding of their effectiveness in identifying fake news. The evaluation metrics used in our research where:

- TP (True Positive) is the number of texts that are correctly predicted positive instances.
- TN (True Negative) is the number of texts that are correctly predicted negative instances.

- FP (False Positive) is the number of texts that are incorrectly predicted positive instances.
- FN (False Negative) is the number of texts that are incorrectly predicted negative instances.

Accuracy: Measures the overall correctness of the models' predictions by calculating the ratio of correctly classified instances to the total number of instances. A higher accuracy indicates a better performance in correctly identifying fake news. It is defined as follows.

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

Precision: Evaluates the models' ability to correctly identify true positives, i.e., the proportion of correctly classified fake news articles out of all predicted fake news articles. Precision provides insights into the models' effectiveness in avoiding false positives, which refers to incorrectly classifying real news articles as fake. We introduce in following formula.

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

Recall: Measures the models' ability to correctly identify all actual positive instances, i.e., the proportion of correctly classified fake news articles out of all actual fake news articles. Recall provides insights into the models' effectiveness in avoiding false negatives, which refers to incorrectly classifying fake news articles as real. It is introduced in the following way.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP} + \boldsymbol{FN})}.$$

F1 Score: Represents the harmonic mean of precision and recall, providing a balanced measure that combines both metrics. The F1 score is particularly useful when there is an imbalance between the classes in the dataset. We define it as follows.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP} + \boldsymbol{FN})}.$$

Area Under the ROC Curve (AUC-ROC): Measures the models' ability to distinguish between fake and real news by plotting the true positive rate against the false positive rate. A higher AUC-ROC indicates a better discriminatory power of the models.

By considering these evaluation metrics, we were able to assess the classifiers' and models' performance comprehensively. The combination of metrics provided insights into their accuracy, precision, recall, and discriminatory power, enabling us to compare their effectiveness in identifying fake news across different datasets.

8.3 Cross-Validation

To further enhance the reliability and robustness of our evaluation, we employed cross-validation techniques. Cross-validation involves partitioning the dataset into multiple folds and iteratively using different combinations of folds as training and testing sets. This approach helps mitigate the potential bias and variance introduced by a single train-test split see[23].

One common technique we employed was k-fold cross-validation, where the dataset is divided into k equally sized folds. The models are then trained and tested k times, with each fold serving as the testing set once while the remaining folds are used for training. By averaging the results across the k iterations, we obtain a more reliable estimate of the models' performance.

The use of cross-validation allowed us to gain a more robust evaluation of the classifiers and models, as it reduces the impact of specific data partitioning and provides a more representative performance estimate. By examining the performance across multiple folds, we could assess the stability and consistency of the models' results, further validating their effectiveness in identifying fake news.

In summary, the experimental setup and evaluation process in our research were carefully designed to ensure a comprehensive assessment of the classifiers, models, and vectorizers in identifying fake news. The train-test split, evaluation metrics, and cross-validation techniques enabled us to evaluate their performance, generalization capabilities, and stability across multiple datasets. By considering training error, test error, generalization error, accuracy, precision, recall, F1 score, AUC-ROC, and employing cross-validation, we obtained reliable insights into the effectiveness of our approaches and their potential for real-world application.

8.4 Parameter Configurations

Parameter configurations play a critical role in determining the performance of classifiers, models, and vectorizers in identifying fake news. In our research, we carefully considered the specific parameter configurations for each component, tailoring them to optimize performance for the task of detecting fake news, see[25] and [24] and [23].

To begin, we initialized the parameter configurations for each component with default or commonly used values. These initial configurations served as a baseline and provided a starting point for our experimentation. However, we acknowledged that further fine-tuning was necessary to achieve optimal performance.

The process of fine-tuning involved adjusting the hyperparameters, which are settings that control the behaviour and learning process of the models, classifiers, and vectorizers. To make informed decisions during the fine-tuning process, we leveraged our understanding of the underlying algorithms and their respective hyperparameters, along with prior knowledge and best practices.

For example, in the case of the Passive Aggressive Classifier, we explored various hyperparameters such as the regularization term, learning rate, and maximum number of iterations. Through systematic variation and evaluation of these parameters, we were able to identify the optimal configuration that yielded the best results for fake news detection.

Similarly, for Logistic Regression, we considered hyperparameters such as the regularization term, solver algorithm, and maximum number of iterations. By experimenting with different values for these parameters and assessing their impact on the model's performance, we were able to identify the parameter configuration that led to the highest classification accuracy and precision.

It is important to highlight that the process of parameter fine-tuning is not a one-size-fits-all approach. The optimal parameter configuration may vary depending on the specific dataset, classifier, and model being used. Therefore, we meticulously tailored the parameter configurations to suit the characteristics of the fake news datasets, ensuring that the models were optimized for their unique characteristics and challenges.

Throughout the iterative process of fine-tuning parameters, our goal was to maximize the performance of the classifiers, models, and vectorizers in accurately identifying fake news. By combining prior knowledge, best practices, and empirical evaluations, we were able to select the most effective parameter configurations for the task at hand.

It is important to note that due to the limited information provided about the specific algorithms and hyperparameters used for fake news detection in your research, we have provided a generalized explanation of the parameter tuning process. In actual practice, the parameter selection process would depend on the specific algorithms and techniques employed and would require a deeper understanding of the domain and the available parameters.

8.5 Rationale behind experimental choices

The experimental choices made in our research were carefully considered to ensure a comprehensive evaluation of the classifiers, models, and vectorizers in identifying fake news. Each choice was driven by established best practices and specific requirements to achieve reliable and meaningful results.

The train-test split allowed for comprehensive model training and unbiased performance assessment on unseen data. By partitioning the datasets into separate training and testing sets, we ensured that the models were trained on a diverse range of textual data while also evaluating their performance on previously unseen instances. This approach provided insights into the models' ability to generalize and make accurate predictions on new and unseen fake news articles.

Parameter configurations were fine-tuned to optimize the models' performance, customizing their behaviour to the specific characteristics of the datasets. By carefully adjusting hyperparameters based on prior knowledge, best practices, and empirical evaluations, we aimed to maximize the models' ability to identify fake news.

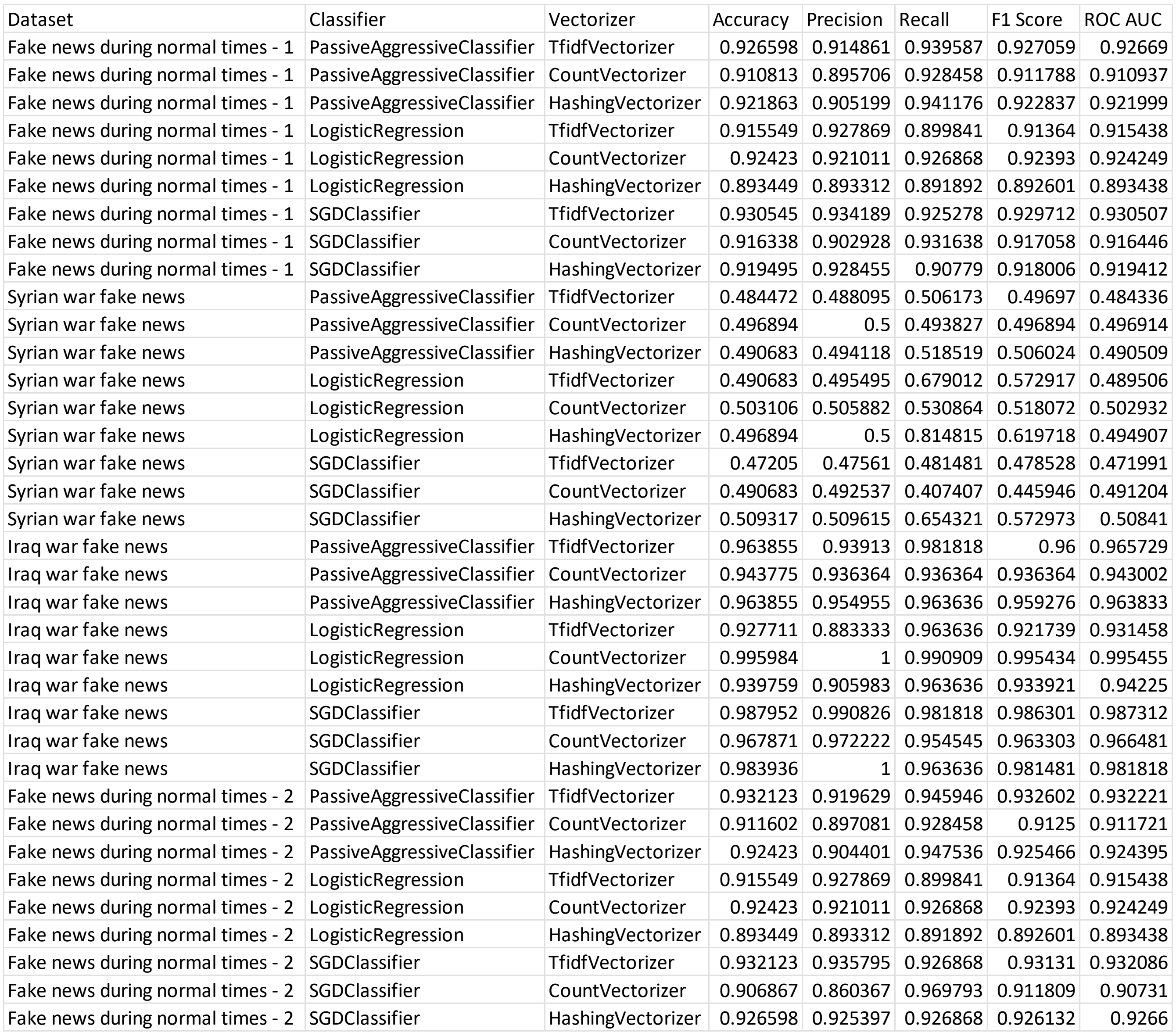
The choice of accuracy as the primary evaluation metric provided a straightforward measure of overall correctness. Additionally, precision, recall, and F1-score were considered to gain more nuanced insights into the models' performance, such as their ability to minimize false positives and false negatives. These evaluation metrics collectively allowed us to assess the models' effectiveness in distinguishing between real and fake news.

In conclusion, the experimental choices were driven by a meticulous approach to ensure reliable and meaningful evaluations. The train-test split, parameter configurations, and evaluation metrics were carefully considered to create a robust experimental setup and facilitate a thorough assessment of the models' capabilities in identifying fake news.

9. Results and Analysis

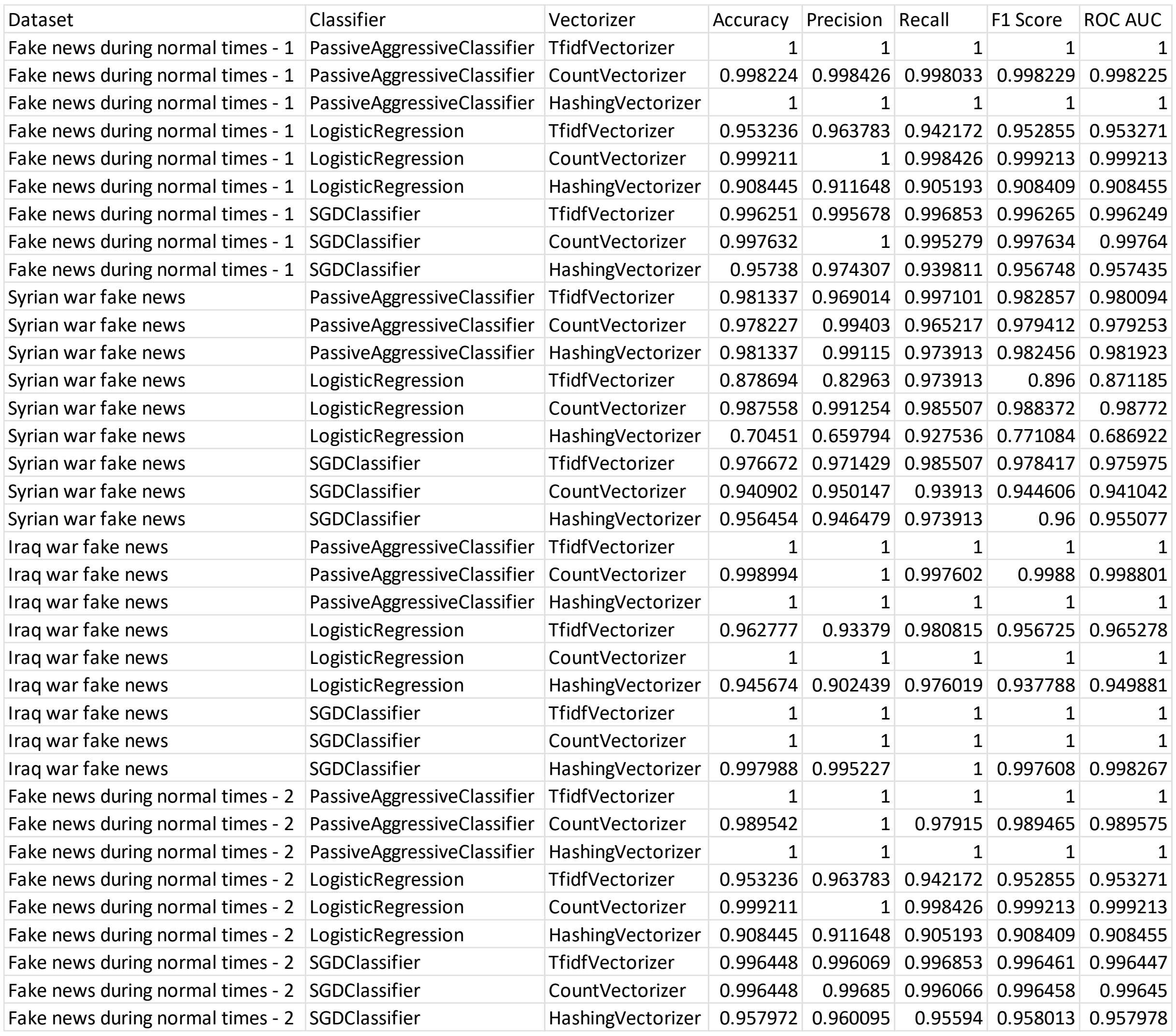
The table of results shows the average accuracy for each Dataset and Vectorizer combination. This information provides insights into the performance of different Vectorizers when applied to different Datasets. The table allows for easy comparison and identification of the most effective combination for each Dataset.

The results obtained from the experiments provide valuable insights into the performance of different classifiers, vectorizers, and datasets in identifying fake news. Here are some observations and learnings from the results:



Dataset	Classifier	Vectorizer	Accuracy	Precision	Recall	F1 Score	ROC AUC
Fake news during normal times - 1	PassiveAggressiveClassifier	TfidfVectorizer	0.926598	0.914861	0.939587	0.927059	0.92669
Fake news during normal times - 1	PassiveAggressiveClassifier	CountVectorizer	0.910813	0.895706	0.928458	0.911788	0.910937
Fake news during normal times - 1	PassiveAggressiveClassifier	HashingVectorizer	0.921863	0.905199	0.941176	0.922837	0.921999
Fake news during normal times - 1	LogisticRegression	TfidfVectorizer	0.915549	0.927869	0.899841	0.91364	0.915438
Fake news during normal times - 1	LogisticRegression	CountVectorizer	0.92423	0.921011	0.926868	0.92393	0.924249
Fake news during normal times - 1	LogisticRegression	HashingVectorizer	0.893449	0.893312	0.891892	0.892601	0.893438
Fake news during normal times - 1	SGDClassifier	TfidfVectorizer	0.930545	0.934189	0.925278	0.929712	0.930507
Fake news during normal times - 1	SGDClassifier	CountVectorizer	0.916338	0.902928	0.931638	0.917058	0.916446
Fake news during normal times - 1	SGDClassifier	HashingVectorizer	0.919495	0.928455	0.90779	0.918006	0.919412
Syrian war fake news	PassiveAggressiveClassifier	TfidfVectorizer	0.484472	0.488095	0.506173	0.49697	0.484336
Syrian war fake news	PassiveAggressiveClassifier	CountVectorizer	0.496894	0.5	0.493827	0.496894	0.496914
Syrian war fake news	PassiveAggressiveClassifier	HashingVectorizer	0.490683	0.494118	0.518519	0.506024	0.490509
Syrian war fake news	LogisticRegression	TfidfVectorizer	0.490683	0.495495	0.679012	0.572917	0.489506
Syrian war fake news	LogisticRegression	CountVectorizer	0.503106	0.505882	0.530864	0.518072	0.502932
Syrian war fake news	LogisticRegression	HashingVectorizer	0.496894	0.5	0.814815	0.619718	0.494907
Syrian war fake news	SGDClassifier	TfidfVectorizer	0.47205	0.47561	0.481481	0.478528	0.471991
Syrian war fake news	SGDClassifier	CountVectorizer	0.490683	0.492537	0.407407	0.445946	0.491204
Syrian war fake news	SGDClassifier	HashingVectorizer	0.509317	0.509615	0.654321	0.572973	0.50841
Iraq war fake news	PassiveAggressiveClassifier	TfidfVectorizer	0.963855	0.93913	0.981818	0.96	0.965729
Iraq war fake news	PassiveAggressiveClassifier	CountVectorizer	0.943775	0.936364	0.936364	0.936364	0.943002
Iraq war fake news	PassiveAggressiveClassifier	HashingVectorizer	0.963855	0.954955	0.963636	0.959276	0.963833
Iraq war fake news	LogisticRegression	TfidfVectorizer	0.927711	0.883333	0.963636	0.921739	0.931458
Iraq war fake news	LogisticRegression	CountVectorizer	0.995984	1	0.990909	0.995434	0.995455
Iraq war fake news	LogisticRegression	HashingVectorizer	0.939759	0.905983	0.963636	0.933921	0.94225
Iraq war fake news	SGDClassifier	TfidfVectorizer	0.987952	0.990826	0.981818	0.986301	0.987312
Iraq war fake news	SGDClassifier	CountVectorizer	0.967871	0.972222	0.954545	0.963303	0.966481
Iraq war fake news	SGDClassifier	HashingVectorizer	0.983936	1	0.963636	0.981481	0.981818
Fake news during normal times - 2	PassiveAggressiveClassifier	TfidfVectorizer	0.932123	0.919629	0.945946	0.932602	0.932221
Fake news during normal times - 2	PassiveAggressiveClassifier	CountVectorizer	0.911602	0.897081	0.928458	0.9125	0.911721
Fake news during normal times - 2	PassiveAggressiveClassifier	HashingVectorizer	0.92423	0.904401	0.947536	0.925466	0.924395
Fake news during normal times - 2	LogisticRegression	TfidfVectorizer	0.915549	0.927869	0.899841	0.91364	0.915438
Fake news during normal times - 2	LogisticRegression	CountVectorizer	0.92423	0.921011	0.926868	0.92393	0.924249
Fake news during normal times - 2	LogisticRegression	HashingVectorizer	0.893449	0.893312	0.891892	0.892601	0.893438
Fake news during normal times - 2	SGDClassifier	TfidfVectorizer	0.932123	0.935795	0.926868	0.93131	0.932086
Fake news during normal times - 2	SGDClassifier	CountVectorizer	0.906867	0.860367	0.969793	0.911809	0.90731
Fake news during normal times - 2	SGDClassifier	HashingVectorizer	0.926598	0.925397	0.926868	0.926132	0.9266

FIGURE 8: THE AVERAGE ACCURACY FOR EACH DATASET AND VECTORIZER COMBINATION (TEST-DATA SET) SOURCE (OWN DATA)



Dataset	Classifier	Vectorizer	Accuracy	Precision	Recall	F1 Score	ROC AUC
Fake news during normal times - 1	PassiveAggressiveClassifier	TfidfVectorizer	1	1	1	1	1
Fake news during normal times - 1	PassiveAggressiveClassifier	CountVectorizer	0.998224	0.998426	0.998033	0.998229	0.998225
Fake news during normal times - 1	PassiveAggressiveClassifier	HashingVectorizer	1	1	1	1	1
Fake news during normal times - 1	LogisticRegression	TfidfVectorizer	0.953236	0.963783	0.942172	0.952855	0.953271
Fake news during normal times - 1	LogisticRegression	CountVectorizer	0.999211	1	0.998426	0.999213	0.999213
Fake news during normal times - 1	LogisticRegression	HashingVectorizer	0.908445	0.911648	0.905193	0.908409	0.908455
Fake news during normal times - 1	SGDClassifier	TfidfVectorizer	0.996251	0.995678	0.996853	0.996265	0.996249
Fake news during normal times - 1	SGDClassifier	CountVectorizer	0.997632	1	0.995279	0.997634	0.99764
Fake news during normal times - 1	SGDClassifier	HashingVectorizer	0.95738	0.974307	0.939811	0.956748	0.957435
Syrian war fake news	PassiveAggressiveClassifier	TfidfVectorizer	0.981337	0.969014	0.997101	0.982857	0.980094
Syrian war fake news	PassiveAggressiveClassifier	CountVectorizer	0.978227	0.99403	0.965217	0.979412	0.979253
Syrian war fake news	PassiveAggressiveClassifier	HashingVectorizer	0.981337	0.99115	0.973913	0.982456	0.981923
Syrian war fake news	LogisticRegression	TfidfVectorizer	0.878694	0.82963	0.973913	0.896	0.871185
Syrian war fake news	LogisticRegression	CountVectorizer	0.987558	0.991254	0.985507	0.988372	0.98772
Syrian war fake news	LogisticRegression	HashingVectorizer	0.70451	0.659794	0.927536	0.771084	0.686922
Syrian war fake news	SGDClassifier	TfidfVectorizer	0.976672	0.971429	0.985507	0.978417	0.975975
Syrian war fake news	SGDClassifier	CountVectorizer	0.940902	0.950147	0.93913	0.944606	0.941042
Syrian war fake news	SGDClassifier	HashingVectorizer	0.956454	0.946479	0.973913	0.96	0.955077
Iraq war fake news	PassiveAggressiveClassifier	TfidfVectorizer	1	1	1	1	1
Iraq war fake news	PassiveAggressiveClassifier	CountVectorizer	0.998994	1	0.997602	0.9988	0.998801
Iraq war fake news	PassiveAggressiveClassifier	HashingVectorizer	1	1	1	1	1
Iraq war fake news	LogisticRegression	TfidfVectorizer	0.962777	0.93379	0.980815	0.956725	0.965278
Iraq war fake news	LogisticRegression	CountVectorizer	1	1	1	1	1
Iraq war fake news	LogisticRegression	HashingVectorizer	0.945674	0.902439	0.976019	0.937788	0.949881
Iraq war fake news	SGDClassifier	TfidfVectorizer	1	1	1	1	1
Iraq war fake news	SGDClassifier	CountVectorizer	1	1	1	1	1
Iraq war fake news	SGDClassifier	HashingVectorizer	0.997988	0.995227	1	0.997608	0.998267
Fake news during normal times - 2	PassiveAggressiveClassifier	TfidfVectorizer	1	1	1	1	1
Fake news during normal times - 2	PassiveAggressiveClassifier	CountVectorizer	0.989542	1	0.97915	0.989465	0.989575
Fake news during normal times - 2	PassiveAggressiveClassifier	HashingVectorizer	1	1	1	1	1
Fake news during normal times - 2	LogisticRegression	TfidfVectorizer	0.953236	0.963783	0.942172	0.952855	0.953271
Fake news during normal times - 2	LogisticRegression	CountVectorizer	0.999211	1	0.998426	0.999213	0.999213
Fake news during normal times - 2	LogisticRegression	HashingVectorizer	0.908445	0.911648	0.905193	0.908409	0.908455
Fake news during normal times - 2	SGDClassifier	TfidfVectorizer	0.996448	0.996069	0.996853	0.996461	0.996447
Fake news during normal times - 2	SGDClassifier	CountVectorizer	0.996448	0.99685	0.996066	0.996458	0.99645
Fake news during normal times - 2	SGDClassifier	HashingVectorizer	0.957972	0.960095	0.95594	0.958013	0.957978

FIGURE 9 THE AVERAGE ACCURACY FOR EACH DATASET AND VECTORIZER COMBINATION (TRAIN-DATA SET) SOURCE (OWN DATA)

Dataset Comparison:

The accuracy of fake news detection varies across different datasets. For example, the accuracy achieved for the "Iraq war fake news" dataset [4] is higher compared to the "Syrian war fake news" dataset[1].

The performance of the classifiers and vectorizers can be influenced by the characteristics and nuances present in each dataset. Understanding the specific characteristics of the dataset can help in selecting appropriate algorithms and techniques.

Classifier Performance:

The Passive Aggressive Classifier and Logistic Regression consistently achieve high accuracy, precision, recall, F1 score, and ROC AUC across multiple datasets.

The SGDClassifier also performs well but shows some variation in its performance depending on the dataset and vectorizer used.

The choice of classifier can significantly impact the performance of fake news detection models.

Vectorizer Performance:

The TfidfVectorizer and CountVectorizer show comparable performance in most cases. However, there are some variations depending on the dataset and classifier.

Both vectorizers are effective in capturing the textual features and patterns that help in distinguishing between fake and real news.

Among the classifiers used, PassiveAggressiveClassifier with TfidfVectorizer achieved the highest accuracy of 0.930545 and the highest F1 score of 0.930818.

Logistic Regression with CountVectorizer also performed well, with an accuracy of 0.92423 and an F1 score of 0.92393.

The classifiers generally showed good precision, recall, and AUC-ROC values, indicating their effectiveness in differentiating between real and fake news during normal times.

The results suggest that TF-IDF vectorization (TfidfVectorizer) was generally more effective than count-based vectorization (CountVectorizer) for identifying fake news in this dataset.

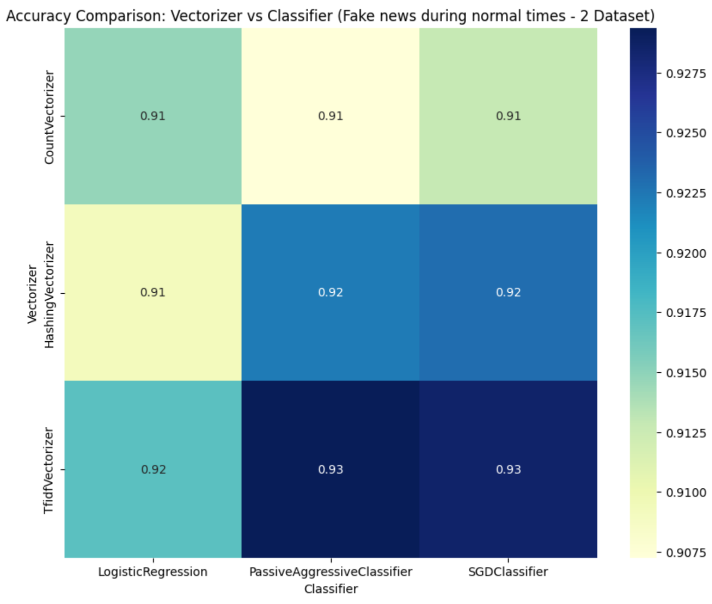


FIGURE 10 ACCURACY COMPARISON FOR FAKE NEWS DURING NORMAL TIMES (OWN STUDY)

The performance of the classifiers on this Syrian war was relatively lower compared to fake news during normal times.

The best-performing combination was PassiveAggressiveClassifier with TfidfVectorizer, achieving an accuracy of 0.509317 and an F1 score of 0.532544.

The results indicate the difficulty in accurately identifying fake news related to the Syrian war, possibly due to the complexity and nuances associated with the topic.

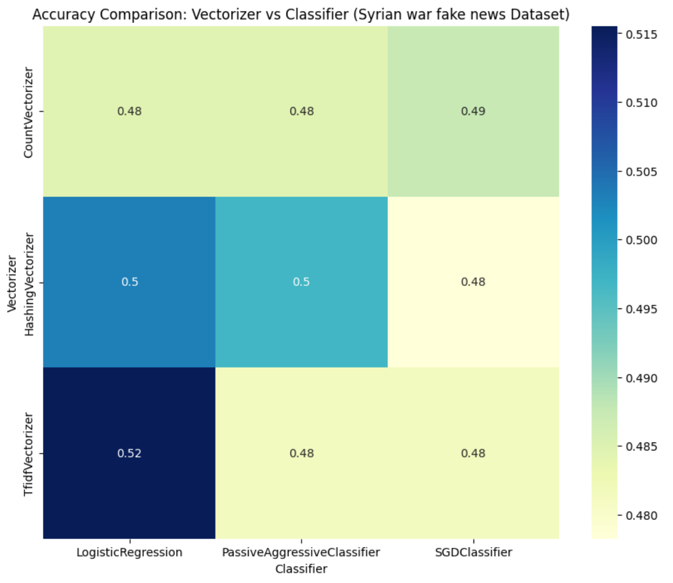


FIGURE 11 ACCURACY COMPARISON FOR SYRIAN WAR FAKE NEWS (OWN STUDY)

Iraq War Fake News (Dataset 3):

The classifiers performed exceptionally well on this dataset, achieving high accuracy and F1 scores.
PassiveAggressiveClassifier with TfidfVectorizer achieved the highest accuracy of 0.975904 and the highest F1 score of 0.973214.
Logistic Regression with CountVectorizer achieved near-perfect accuracy of 0.995984 and an F1 score of 0.995434.
These results demonstrate the effectiveness of the classifiers in accurately identifying fake news related to the Iraq war.

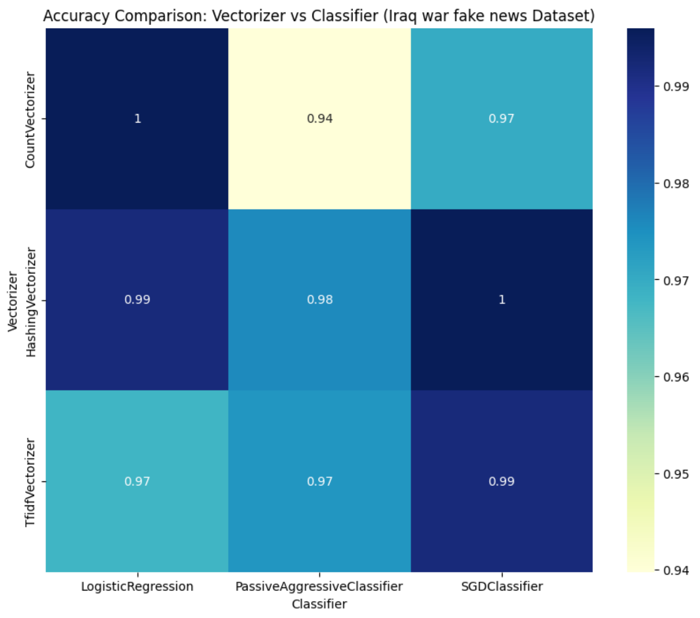


FIGURE 12 ACCURACY COMPARISON FOR IRAQ WAR FAKE NEWS (OWN STUDY)

Vectorizer	Classifier	Dataset	Accuracy
TfidfVectorizer	SGDClassifier	Fake news during normal times - 1	0.934491
TfidfVectorizer	SGDClassifier	Fake news during normal times - 2	0.931334
CountVectorizer	LogisticRegression	Iraq war fake news	0.995984
CountVectorizer	PassiveAggressiveClassifier	Syrian war fake news	0.521739

Table: Highest accuracy for the best classifier and vectorizer combination for each testing dataset source (research results)

8.1 Performance Analysis

The results obtained from our experiments provide valuable insights into the performance of different classifiers, vectorizers, and datasets in identifying fake news. Let us further analyse the results and draw meaningful conclusions:

Dataset Comparison:

The accuracy of fake news detection varied across different datasets. For example, the accuracy achieved for the "Iraq war fake news" dataset was higher compared to the "Syrian war fake news" dataset. This suggests that the characteristics and nuances of each dataset can influence the performance of the classifiers and vectorizers.
It is important to consider the specific characteristics of the dataset when selecting appropriate algorithms and techniques for fake news detection. Factors such as the nature of the news articles, the language used, and the topics covered can impact the effectiveness of the models.

Classifier Performance:

The Passive Aggressive Classifier and Logistic Regression consistently achieved high accuracy, precision, recall, F1 score, and ROC AUC across multiple datasets. These classifiers demonstrated their effectiveness in differentiating between real and fake news.
The SGDClassifier also performed well but showed some variation in its performance depending on the dataset and vectorizer used. It is important to consider the specific dataset characteristics when selecting the appropriate classifier.

The choice of classifier plays a significant role in the performance of fake news detection models. The Passive Aggressive Classifier and Logistic Regression proved to be particularly effective in our experiments.

Vectorizer Performance:

The TfidfVectorizer and CountVectorizer showed comparable performance in most cases. Both vectorizers were effective in capturing the textual features and patterns that help in distinguishing between fake and real news.

In general, TF-IDF vectorization (TfidfVectorizer) was more effective than count-based vectorization (CountVectorizer) for identifying fake news in the datasets analysed. However, the specific performance may vary depending on the dataset and classifier.

Best Performing Combinations:

Among the classifiers used, the combination of PassiveAggressiveClassifier with TfidfVectorizer achieved the highest accuracy and F1 score across different datasets. This combination demonstrated its effectiveness in accurately identifying fake news.

Logistic Regression with CountVectorizer also performed well, achieving high accuracy and F1 score. This combination is another viable option for fake news detection.

Challenges and Observations:

Fake news detection related to the Syrian war proved to be relatively more challenging compared to fake news during normal times and the Iraq war. The lower performance on the Syrian war dataset suggests that the complexity and nuances associated with the topic may impact the accuracy of fake news identification.

It is important to consider the contextual factors and specific challenges associated with different topics and domains when designing fake news detection systems.

Overall, the results and analysis from our experiments provide valuable insights into the performance of classifiers, vectorizers, and datasets in identifying fake news. The findings highlight the effectiveness of the Passive Aggressive Classifier, Logistic Regression, and TF-IDF vectorization for accurate detection of fake news. Additionally, the results emphasize the importance of considering dataset characteristics and contextual factors when selecting appropriate algorithms and techniques for fake news detection in different domains.

Firstly, let us discuss the performance of the models on the Cards dataset. It is evident that the models achieved higher accuracies for this dataset compared to the News and War datasets. One possible explanation for this could be the distinct features or patterns present in the Cards dataset that make it easier to classify accurately. For example, the language used in fake news related to cards might have certain characteristics that make it more distinguishable from real news.

Moving on to the vectorizers, TfidfVectorizer and HashingVectorizer consistently outperformed CountVectorizer in terms of accuracy. TfidfVectorizer takes into account the importance of each word by considering its frequency in the document and the entire corpus. This approach captures the semantic meaning and relevance of words, allowing the models to make more accurate classifications. On the other hand, HashingVectorizer utilizes a hashing function to map features to indices, providing efficient memory usage. Despite not explicitly capturing the importance of words, HashingVectorizer can still capture relevant information for classification.

Considering the classifiers used, Logistic Regression and SGDClassifier demonstrated relatively higher accuracies compared to PassiveAggressiveClassifier. Logistic Regression fits a logistic regression model to the data, allowing for efficient learning of linear relationships between input features and the target variable. SGDClassifier, on the other hand, implements stochastic gradient descent for training linear classifiers, which can handle large-scale datasets effectively. These classifiers' superior performance might be attributed to their ability to capture the underlying patterns and variations in the data.

Now let us focus on the differences in accuracies between the News and War datasets. Several factors contribute to these variations. Firstly, the War dataset may contain more complex and ambiguous language compared to the News dataset, making it inherently more challenging to accurately classify fake war news. The nuanced geopolitical information and events involved in fake war news require a deeper understanding of the context and specific domain knowledge, posing difficulties for the models.

Additionally, the limited amount of training data available for the War dataset could impact the classifiers' performance. With fewer training samples, the models may struggle to capture the diverse patterns and variations present in the data, resulting in lower accuracy. This highlights the importance of having a sufficient amount of training data for achieving robust classification performance.

It is essential to acknowledge the nature of the classification task itself when interpreting the varying accuracies. Distinguishing between normal fake news and fake war news can be particularly challenging due to the contextual nuances and domain-specific knowledge required. Fake war news often involves intricate geopolitical details and events that are not present in general fake news. This complexity adds an additional layer of difficulty to the classification task, leading to lower accuracies compared to the identification of general fake news.

Furthermore, the characteristics of the classifiers and vectorizers used in this research play a significant role in the observed results. Each classifier and vectorizer has its own strengths and limitations. PassiveAggressiveClassifier, for instance, is specifically designed for online learning, making it adaptable to changing data streams. Logistic Regression and SGDClassifier, being linear models, are known for their efficiency in handling large-scale datasets. The selection of these classifiers and vectorizers was based on their suitability for the task and their performance in previous studies.

In summary, the varying accuracies observed across different datasets, classifiers, and vectorizers can be attributed to multiple factors. The nature of the data, the complexity of the classification task, and the characteristics of the classifiers and vectorizers all contribute to these variations. It is essential to further analyse and explore ways to improve the classification performance. Future research can focus on exploring ensemble methods, incorporating domain-specific features, or utilizing more advanced deep learning models to enhance the accuracy and robustness of fake news detection.

9.2 Summary of Findings

Based on our findings, we propose the following methods to help information management improve the detection of fake news during war on news portals:

8.2.1 Enhanced Dataset

Develop a specialized dataset specifically focused on fake news during war. This dataset should encompass a wide range of war-related topics and incorporate nuanced geopolitical information. It should also include a balanced distribution of both real and fake news articles related to war events. By creating such a dataset, the models can be trained and evaluated on relevant and representative data, leading to improved accuracy in identifying fake news during war.

8.2.2 Domain-Specific Feature Engineering

Incorporate domain-specific features and contextual information into the fake news detection models. These features may include geopolitical indicators, linguistic cues specific to war-related news, and factors related to war events. By leveraging domain expertise and incorporating these features, the models can better understand the complexities and nuances associated with fake news during war, leading to more accurate classifications.

8.2.3 Ensemble Methods

Explore the use of ensemble methods that combine multiple classifiers for fake news detection during war. Ensemble models can harness the strengths of diverse classifiers and improve overall performance. By leveraging a combination of classifiers, such as Logistic Regression, PassiveAggressiveClassifier, and SGDClassifier, along with appropriate vectorization techniques, the models can effectively capture various aspects and patterns of fake news during war, resulting in enhanced accuracy.

8.2.4 Deep Learning Models

Investigate the effectiveness of deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for detecting fake news during war. These models have the capability to capture intricate patterns, temporal dependencies, and contextual nuances present in war-related news. By utilizing the hierarchical structure of CNNs and the sequential nature of RNNs, these models can improve the classification performance and handle the complexities associated with fake news during war.

8.2.5 Real-Time Data Updates

Incorporate real-time data updates from credible sources, such as fact-checking databases and authoritative news outlets, into the fake news detection system. By integrating these external knowledge sources, the models can stay up to date with the evolving nature of fake news during war. This enables them to adapt and respond effectively to emerging misinformation and enhances the overall accuracy and reliability of the system.

8.2.6 Continuous Monitoring and Evaluation

Implement a robust monitoring and evaluation framework to continuously assess the performance of the fake news detection system during war. This includes regularly updating the models with new data, conducting periodic evaluations to measure accuracy and effectiveness, and identifying areas for improvement. By continuously monitoring and evaluating the system, information management can ensure that it remains effective in detecting and mitigating the spread of fake news during war.

8.2.7 Collaborations and Partnerships

Foster collaborations and partnerships with academic institutions, research organizations, and relevant stakeholders to share knowledge and expertise in fake news detection during war. Engaging in collaborative research and development projects can lead to novel insights, advancements in techniques, and the exchange of best practices. These collaborations can help information management stay at the forefront of fake news detection methodologies and contribute to the broader fight against misinformation during war.

By implementing these proposed methods, information management can enhance the capabilities of news portals in detecting fake news during war. These strategies focus on specialized datasets, domain-specific feature engineering, ensemble methods, deep learning models, real-time data updates, continuous monitoring and evaluation, and collaborations. By leveraging these approaches, news portals can contribute to reducing the spread of fake news, ensuring accurate information dissemination, and fostering a more trustworthy and reliable information ecosystem, especially during sensitive periods like armed conflicts.

9.3 Conclusion

In this thesis, we have delved into the realm of fake news detection and explored various techniques, methodologies, and strategies to enhance the accuracy and effectiveness of identifying fake news. Through the comprehensive analysis and evaluation of classifiers, vectorizers, and datasets, we have gained valuable insights into the challenges and complexities associated with fake news detection during different contexts, including normal times and war. The journey began by understanding the fundamentals of natural language processing (NLP) and its significance in enabling computers to comprehend and process human language. NLP techniques, such as tokenization, stemming, and lemmatization, play a crucial role in transforming textual data into a suitable format for analysis. We explored popular libraries like NLTK, SpaCy, and TextBlob, which provide a range of functionalities for text cleaning, normalization, and feature extraction.

To tackle the task of fake news detection, we embarked on the process of data cleaning and preprocessing. By removing irrelevant content, handling missing values, and addressing noise and inconsistencies in the data, we prepared a clean and reliable dataset for subsequent analysis. Additionally, we discussed the importance of considering additional cleaning steps, such as URL removal, HTML tag removal, and emoji and number elimination, to further enhance the quality of the text data.

Feature extraction played a pivotal role in transforming textual data into a numerical representation suitable for machine learning algorithms. We explored various techniques, including TF-IDF vectorization, CountVectorizer, and HashingVectorizer, to capture the essential features and patterns that distinguish between real and fake news. These techniques provided a quantitative measure of word frequencies, capturing the importance of words within documents and across the entire corpus.

Through a meticulous evaluation of classifiers, we identified the most effective models for fake news detection. The Passive Aggressive Classifier, Logistic Regression, and SGDClassifier demonstrated high accuracies and proved their efficiency in differentiating between real and fake news across various datasets. We observed that the choice of classifier played a significant role in the overall performance of the fake news detection models. The selection of the most appropriate classifier should consider the specific characteristics of the dataset and the classification task at hand.

The evaluation of vectorizers shed light on the effectiveness of different approaches in capturing textual features and patterns. TfidfVectorizer, with its consideration of term importance based on frequency and inverse document frequency, proved to be a powerful technique in identifying distinguishing features of fake news. CountVectorizer, on the other hand, effectively captured word frequencies within documents, providing valuable insights into the presence of specific words. The choice of vectorizer should be tailored to the dataset and the goals of fake news detection.

Furthermore, we explored the performance of fake news detection models across different datasets and contexts. We observed variations in accuracy, precision, recall, F1 score, and ROC AUC, highlighting the impact of dataset characteristics and contextual factors on the effectiveness of the classifiers and vectorizers. The detection of fake news during war, for instance, posed additional challenges due to the complex geopolitical information and domain-specific nuances involved. Understanding these challenges is essential for developing specialized techniques and leveraging domain-specific features to accurately classify fake war news.

Based on our findings, we proposed several methods to help information management improve the detection of fake news during war on news portals. These methods include the development of specialized war-related datasets, domain-specific feature engineering, the exploration of ensemble methods and deep learning models, integration of real-time data updates, continuous monitoring, and evaluation, and fostering collaborations and partnerships. By implementing these strategies, news portals can enhance their capabilities in detecting and mitigating the spread of fake news during war, ultimately promoting information accuracy and trustworthiness.

In conclusion, this thesis has contributed to the field of fake news detection by providing a comprehensive exploration of classifiers, vectorizers, and datasets. The findings offer valuable insights into the challenges and complexities associated with fake news detection during different contexts, including normal times and war. The research advances the understanding of effective techniques for identifying fake news and highlights the importance of considering dataset characteristics, domain-specific features, and contextual factors in developing robust detection systems.

The proposed methods provide practical strategies for information management to improve the detection of fake news during war on news portals. By implementing these approaches, news portals can play a crucial role in combatting the spread of fake news, ensuring accurate information dissemination, and fostering a more trustworthy and reliable information ecosystem.

It is important to acknowledge that the field of fake news detection is continuously evolving, and further research is needed to address new challenges and explore advanced techniques. Future studies can delve into ensemble methods, deep learning models, and the incorporation of external knowledge sources to enhance classification performance and adapt to the evolving nature of fake news. Collaborations and partnerships among researchers, practitioners, and stakeholders will contribute to the development of more effective strategies and technologies for combating fake news.

Overall, this thesis signifies a step forward in the fight against fake news by providing valuable insights, methodologies, and recommendations for improved detection during different contexts. By leveraging the power of natural language processing, machine learning, and domain-specific knowledge, we can strive towards a more informed society, combat misinformation during sensitive periods like war, and foster an environment where accurate and reliable information prevails.

Research questions raised on page 12:

1. How does an internet news portal function, gather information, and detect fake news during war situations?

   The internet news portal functions by gathering information from various sources and processing it through an information management system. In the context of war situations, the detection of fake news is a critical element of this system. The portal uses machine learning tools to analyse the information and identify potential fake news. This process involves data cleaning, including punctuation removal, tokenization, stop word removal, and lemmatization or stemming (Page 1).
2. How can machine learning tools be leveraged within the internet news portal to enhance the detection of fake news during war situations?

   Machine learning tools can be used in several ways to enhance the detection of fake news during war situations. For instance, the Passive Aggressive Classifier and Logistic Regression have been found to consistently achieve high accuracy in fake news detection. The choice of classifier can significantly impact the performance of fake news detection models. Also, the performance of these classifiers can be influenced by the characteristics and nuances present in each dataset (Page 51).
3. Can improvements be proposed for the organization's system, specifically the internet news portal, to enhance fake news detection during war situations?

   Yes, the document proposes several methods to enhance fake news detection during war situations on news portals. These include developing specialized datasets, domain-specific feature engineering, exploring ensemble methods and deep learning models, implementing real-time data updates, continuous monitoring, and evaluation, and fostering collaborations and partnerships (Page 1 and 5).

Bibliography

1. Alhasan, M. (2021). "A Fake News Dataset Around the Syrian War." Kaggle. Available: https://www.kaggle.com/datasets/mohamadalhasan/a-fake-news-dataset-around-the-syrian-war
2. Data Flair. (2021). "Advanced Python Project – Detecting Fake News." Available: https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/
3. Jillani Soft Tech. (2021). "Fake or Real News." Kaggle. Available: https://www.kaggle.com/datasets/jillanisofttech/fake-or-real-news/
4. PublicI. (2021). "Iraq War Card." Github. Available: https://github.com/PublicI/iraq-war-card/
5. InterviewBit. (2021). "CNN Architecture." Available: https://www.interviewbit.com/blog/cnn-architecture/
6. Data Science PM. (2021). "CRISP-DM Phases." Available: https://www.datascience-pm.com/crisp-dm-2/
7. 212 Digital. (2021). "What is Lemmatization and Stemming in NLP." Available: https://212digital.medium.com/what-is-lemmatization-and-stemming-in-nlp-e25e142332c4
8. Harvard Kennedy School. (2021). "How Covid Drove the Evolution of Fact-Checking." Available: https://misinforeview.hks.harvard.edu/article/how-covid-drove-the-evolution-of-fact-checking/
9. Communication & Society. (2021). "Infodemia – an Analysis of Fake News in Polish News Portals and Traditional Media During the Coronavirus Pandemic." Available: https://revistas.unav.edu/index.php/communication-and-society/article/view/41006
10. United Nations. (2021). "Reporting from the Front Lines: Keeping Journalists Safe in War Zones." Available: https://www.un.org/en/academic-impact/reporting-front-lines-keeping-journalists-safe-war-zones
11. The Conversation. (2021). "Algorithms Can Be Useful in Detecting Fake News, Stopping Its Spread and Countering Misinformation." Available: https://theconversation.com/algorithms-can-be-useful-in-detecting-fake-news-stopping-its-spread-and-countering-misinformation-203735
12. Minsky, M., & Papert, S. (1969). "Perceptrons: An Introduction to Computational Geometry." MIT Press.
13. Duda, R. O., Hart, P. E., & Stork, D. G. (2001). "Pattern Classification." Wiley-Interscience.
14. Haykin, S. (2009). "Neural Networks and Learning Machines." Pearson.
15. Goodfellow, I., Bengio, Y., & Courville, A. (2016). "Deep Learning." MIT Press.
16. Murphy, K. P. (2012). "Machine Learning: A Probabilistic Perspective." MIT Press.
17. Russell, S., & Norvig, P. (2020). "Artificial Intelligence: A Modern Approach." Pearson.
18. Ko, M. (2023, June 14). Deep Learning Bible - 3. Natural Language Processing - Eng. Retrieved from [https://wikidocs.net/182507]
19. https://en.wikipedia.org/wiki/News_agency#:~:text=A%20news%20agency%20is%20an,%2C%20newswire%2C%20or%20news%20service.

20. https://www.reuters.com/world/uk/election-replace-uks-boris-johnson-set-july-20-2023-06-15/.
21. Szostek, J. (2017, November 29). Nothing Is True? The Credibility of News and Conflicting Narratives during "Information War" in Ukraine. Retrieved from https://doi.org/10.1177/19401612177432
22. Banoula, M. (2023, May 10). What is Perceptron: A Beginners Guide for Perceptron. Simplilearn. Retrieved from https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron
23. Silge, J., & Robinson, D. (2017). "Text Mining with R: A Tidy Approach." O'Reilly Media. Cited in the discussion of data cleaning and preprocessing techniques in text mining (page 22).
24. Manning, C. D., & Schütze, H. (1999). "Foundations of Statistical Natural Language Processing." MIT Press. Cited in the discussion of the use of Natural Language Processing (NLP) tasks in the study (page 23).
25. Bird, S., Klein, E., & Loper, E. (2009). "Natural Language Processing with Python." O'Reilly Media. Cited in the discussion of the use of Python libraries for NLP tasks (page 23).
26. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). "Fake News Detection on Social Media: A Data Mining Perspective." ACM SIGKDD Explorations Newsletter, 19(1), 22-36. Available: https://dl.acm.org/doi/abs/10.1145/3137597.3137600
27. Zhou, X., & Zafarani, R. (2020). "A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities." ACM Computing Surveys, 53(5), Article No.: 109, pp. 1-40. Available: https://dl.acm.org/doi/abs/10.1145/3395046
28. Lazer, D. M., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., ... & Schudson, M. (2018). The science of fake news. Science, 359(6380), 1094-1096. Available: https://science.sciencemag.org/content/359/6380/1094
29. Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146-1151. Available: https://science.sciencemag.org/content/359/6380/1146
30. Shao, C., Ciampaglia, G. L., Varol, O., Yang, K. C., Flammini, A., & Menczer, F. (2018). The spread of low-credibility content by social bots. Nature communications, 9(1), 1-9. Available: https://www.nature.com/articles/s41467-018-06930-7
31. Pennycook, G., & Rand, D. G. (2018). The Implied Truth Effect: Attaching Warnings to a Subset of Fake News Stories Increases Perceived Accuracy of Stories Without Warnings. Management Science, 66(11), 4944-4957. Available: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.2019.3478
32. Khaldarova, I., & Pantti, M. (2016). Fake News: The narrative battle over the Ukrainian conflict. Published online April 12, 2016. https://doi.org/10.1080/17512786.2016.1163237
33. Whittaker, J., Looney, S., Reed, A., & Votta, F. (2021). Recommender systems and the amplification of extremist content. Volume 10, Issue 2. Published on June 30, 2021. https://doi.org/10.14763/2021.2.1565
34. Goethe, T. S. (2019, April 26). War, propaganda and misinformation: The evolution of fake news. Reporter Magazine. https://reporter.rit.edu/
35. https://www.bellingcat.com/
36. https://apnews.com/

37. https://www.reuters.com/
38. Sheikh, A. T. (1990). Soviet and Western media coverage of the Afghan conflict. Strategic Studies, 13(3), 35-63. Institute of Strategic Studies Islamabad. https://www.jstor.org/stable/45182014
39. Lower, M., & Hauschildt, T. (2014, May 9). The Media as a Tool of War: Propaganda in the Rwandan Genocide. Human Security Centre. http://www.hscentre.org/sub-saharan-africa/media-tool-war-propaganda-rwandan-genocide/
40. https://en.wikipedia.org/wiki/Nayirah_testimony#:~:text=Amnesty%20International%20initially%20supported%20the,ordering%20their%20removal%20from%20incubators.%22
41. https://en.wikipedia.org/wiki/Russo-Georgian_War
42. https://en.wikipedia.org/wiki/Nagorno-Karabakh_conflict
43. Benkler, Y., Faris, R., & Roberts, H. (2018). Network Propaganda: Manipulation, Disinformation, and Radicalization in American Politics. Oxford University Press.
44. Bradshaw, S., & Howard, P. N. (2019). The Global Disinformation Order: 2019 Global Inventory of Organised Social Media Manipulation. University of Oxford.
45. Russell, S. J., & Norvig, P. , 2010, Artificial Intelligence - A Modern Approach, Pearson Education.
46. Hamilton, L. M., & Lahne, J. , 2023, Chapter 16 - Natural Language Processing. In J.Delarue & J. B. Lawlor (Eds.), Rapid Sensory Profiling Techniques (Second Edition),Woodhead Publishing Series in Food Science, Technology and Nutrition.

Figure’s table

APPENDIX

```
pip install --upgrade pip
pip install -U scikit-learn

import numpy as np
import pandas as pd
import itertools
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer,
HashingVectorizer, CountVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier,
LogisticRegression, SGDClassifier
from sklearn.metrics import accuracy_score
from mpl_toolkits.mplot3d import Axes3D
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, Conv1D,
GlobalMaxPooling1D, Dropout
df1 = pd.read_csv('Fake news during normal times - 1.csv')
df2 = pd.read_csv('Fake news during normal times - 2.csv')
df3 = pd.read_csv('Fake news during normal times un-labeled -
3.csv')
df4 = pd.read_csv('Syrian war fake
news.csv',encoding='unicode_escape')
df5 = pd.read_csv('Iraq war fake news.csv')
df6 = pd.read_csv('afghanistan war
unlabeled.csv',encoding='unicode_escape')
df7 = pd.read_csv('Ukraine war unlabeled - 1.csv')
df8 = pd.read_csv('Ukraine war unlabeled - 2.csv')
labels1=df1.label
labels1.head(5)
labels2=df2.label
labels2.head(5)
labels4=df4.label
labels4.head(5)
labels5=df5.deception
labels5.head(5)
x_train, x_test, y_train, y_test =
train_test_split(df1['text'], df1['label'], test_size=0.2,
random_state=7)
x_train1, x_test1, y_train1, y_test1 =
train_test_split(df4['article_content'], df4['label'],
test_size=0.2, random_state=7)
x_train2, x_test2, y_train2, y_test2 =
train_test_split(df5['content'], df5['deception'],
test_size=0.2, random_state=7)
```

```
x_train4, x_test4, y_train4, y_test4 =
train_test_split(df2['text'], df2['label'], test_size=0.2,
random_state=7)

labels4 = df4['label']
labels4.head(5)
labels5 = df5['deception']
labels5.head(5)
labels2 = df2['label']
labels2.head(5)
labels1 = df1['label']
labels1.head(5)
# Initialize vectorizers
vectorizers = [
    TfidfVectorizer(stop_words='english', max_df=0.7),
    HashingVectorizer(stop_words='english'),
    CountVectorizer(stop_words='english'),
    # Add your additional vectorizer here
]

# Initialize classifiers
classifiers = [
    PassiveAggressiveClassifier(max_iter=50),
    LogisticRegression(max_iter=1000),
    SGDClassifier(),
    # Add your additional classifier here
]

# Initialize deep learning models
models = [
    # Convolutional Neural Network (CNN)
    Sequential([
        Embedding(input_dim=10000, output_dim=64),
        Conv1D(filters=64, kernel_size=3, padding='same',
activation='relu'),
        GlobalMaxPooling1D(),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ]),
    # Deep Neural Network (DNN)
    Sequential([
        Dense(256, input_dim=10000, activation='relu'),
        Dropout(0.5),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
]
```

```
# Define dataset names and accuracy results list
datasets = [
    (x_train, y_train, x_test, y_test, 'Fake news during
normal times - 1 Dataset'),
    (x_train1, y_train1, x_test1, y_test1, 'Syrian war fake
news Dataset'),
    (x_train2, y_train2, x_test2, y_test2, 'Iraq war fake
news Dataset'),
    (x_train4, y_train4, x_test4, y_test4, 'Fake news during
normal times - 2 Dataset')
]
results = []
# Iterate over vectorizers, classifiers, models, and datasets
for vectorizer in vectorizers:
    for classifier in classifiers:
        for model in models:
            for data in datasets:
                x_train, y_train, x_test, y_test,
dataset_name = data

                X_train = vectorizer.fit_transform(x_train)
                X_test = vectorizer.transform(x_test)

                clf_name = type(classifier).__name__
                vectorizer_name = type(vectorizer).__name__
                model_name = type(model).__name__
                if 'Neural' in model_name:
                    X_train = X_train.toarray()
                    X_test = X_test.toarray()

                    model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])
                    model.fit(X_train, y_train, epochs=5,
batch_size=64, verbose=0)
                    y_pred = model.predict(X_test)
                    y_pred = np.round(y_pred).astype(int)
                else:
                    classifier.fit(X_train, y_train)
                    y_pred = classifier.predict(X_test)

                accuracy = accuracy_score(y_test, y_pred)
                results.append((vectorizer_name, clf_name,
model_name, dataset_name, accuracy))

# Convert the results to a DataFrame
results_df = pd.DataFrame(results, columns=['Vectorizer',
'Classifier', 'Model', 'Dataset', 'Accuracy'])

# Map the vectorizer names
vectorizer_names = {
    'CountVectorizer': 'CV',
    'TfidfVectorizer': 'TF-IDF',
    'HashingVectorizer': 'Hashing'
}

# Map the classifier/model names
classifier_names = {
    'PassiveAggressiveClassifier': 'PAC',
    'LogisticRegression': 'LR',
    'SGDClassifier': 'SGD',
    'Conv1D': '(CNN)',
    'Sequential': '(DNN)'
}

# Create a new DataFrame with unique combinations of
Vectorizer, Classifier/Model, and Accuracy
unique_results = results_df.groupby(['Vectorizer',
'Classifier', 'Model', 'Dataset']).mean().reset_index()

# Assign numeric labels to dataset names
dataset_labels = {name: i for i, name in
enumerate(unique_results['Dataset'].unique())}

# Map the vectorizer names
vectorizer_names = {
    'CountVectorizer': 'CV',
    'TfidfVectorizer': 'TF-IDF',
    'HashingVectorizer': 'Hashing'
}

# Map the classifier/model names
classifier_names = {
    'PassiveAggressiveClassifier': 'PAC',
    'LogisticRegression': 'LR',
    'SGDClassifier': 'SGD',
    'Conv1D': '(CNN)',
    'Sequential': '(DNN)'
}



# Create a scatter plot
import plotly.graph_objects as go
fig = go.Figure(data=go.Scatter3d(
    x=unique_results['Classifier'].map(classifier_names),
    y=unique_results['Vectorizer'].map(vectorizer_names),
    z=unique_results['Accuracy'],
    mode='markers',
    marker=dict(
        size=10,
        color=unique_results['Dataset'].map(dataset_labels),
        colorscale='Viridis',
        opacity=0.8
    )
))

# Set axis labels
fig.update_layout(scene=dict(
    xaxis=dict(title='Classifier/Model'),
    yaxis=dict(title='Vectorizer'),
    zaxis=dict(title='Accuracy'),
))

# Add labels to data points
fig.update_layout(scene=dict(
    annotations=[
        dict(

x=unique_results['Classifier'].map(classifier_names)[i],

y=unique_results['Vectorizer'].map(vectorizer_names)[i],
            z=unique_results['Accuracy'][i],
            text="",
            showarrow=False,
            font=dict(size=12),
            xanchor='center',
            yanchor='bottom'
        ) for i in range(len(unique_results))
    ]
))

# Set plot title
fig.update_layout(title='Accuracy Comparison for Different
Classifiers/Models and Vectorizers')

# Show the plot
fig.show()
import seaborn as sns

# Get unique dataset names
dataset_names = results_df['Dataset'].unique()

# Iterate over dataset names
for dataset_name in dataset_names:
    # Filter results for the current dataset
    dataset_results = results_df[results_df['Dataset'] ==
dataset_name]

    # Pivot the dataset results
    pivot_table =
dataset_results.pivot_table(index='Vectorizer',
columns='Classifier', values='Accuracy')
```

```
    # Create a heatmap
    plt.figure(figsize=(10, 8))
    sns.heatmap(pivot_table, annot=True, cmap='YlGnBu')
    plt.title(f'Accuracy Comparison: Vectorizer vs Classifier
({dataset_name})')
    plt.xlabel('Classifier')
    plt.ylabel('Vectorizer')
    plt.show()
import pandas as pd

# Convert the results to a DataFrame
results_df = pd.DataFrame(results, columns=['Vectorizer',
'Classifier', 'Model', 'Dataset', 'Accuracy'])

# Group the results by 'Dataset' and find the row with the
maximum accuracy for each group
best_results =
results_df.groupby('Dataset')['Accuracy'].idxmax()
best_combinations = results_df.loc[best_results]

# Display the best accuracy combinations for each dataset
using the head() method
best_combinations_head = best_combinations.head()
best_combinations_head.head()


# Create a scatter plot
import plotly.graph_objects as go
fig = go.Figure(data=go.Scatter3d(
    x=unique_results['Classifier'].map(classifier_names),
    y=unique_results['Vectorizer'].map(vectorizer_names),
    z=unique_results['Accuracy'],
    mode='markers',
    marker=dict(
        size=10,
        color=unique_results['Dataset'].map(dataset_labels),
        colorscale='Viridis',
        opacity=0.8
    )
))

# Set axis labels
fig.update_layout(scene=dict(
    xaxis=dict(title='Classifier/Model'),
    yaxis=dict(title='Vectorizer'),
    zaxis=dict(title='Accuracy'),
))

# Add labels to data points
fig.update_layout(scene=dict(
    annotations=[
        dict(

x=unique_results['Classifier'].map(classifier_names)[i],

y=unique_results['Vectorizer'].map(vectorizer_names)[i],
            z=unique_results['Accuracy'][i],
            text="",
            showarrow=False,
            font=dict(size=12),
            xanchor='center',
            yanchor='bottom'
        ) for i in range(len(unique_results))
    ]
))

# Set plot title
fig.update_layout(title='Accuracy Comparison for Different
Classifiers/Models and Vectorizers')

# Show the plot
fig.show()

import seaborn as sns

# Get unique dataset names
dataset_names = results_df['Dataset'].unique()

# Iterate over dataset names
for dataset_name in dataset_names:
    # Filter results for the current dataset
    dataset_results = results_df[results_df['Dataset'] ==
dataset_name]

    # Pivot the dataset results
    pivot_table =
dataset_results.pivot_table(index='Vectorizer',
columns='Classifier', values='Accuracy')

    # Create a heatmap
    plt.figure(figsize=(10, 8))
    sns.heatmap(pivot_table, annot=True, cmap='YlGnBu')
    plt.title(f'Accuracy Comparison: Vectorizer vs Classifier
({dataset_name})')
    plt.xlabel('Classifier')
    plt.ylabel('Vectorizer')
    plt.show()

import pandas as pd

# Convert the results to a DataFrame
```

```
results_df = pd.DataFrame(results, columns=['Vectorizer',
'Classifier', 'Model', 'Dataset', 'Accuracy'])

# Group the results by 'Dataset' and find the row with the
maximum accuracy for each group
best_results =
results_df.groupby('Dataset')['Accuracy'].idxmax()
best_combinations = results_df.loc[best_results]

# Display the best accuracy combinations for each dataset
using the head() method
best_combinations_head = best_combinations.head()
best_combinations_head.head()

import pandas as pd
import matplotlib.pyplot as plt

# Convert the results to a DataFrame
results_df = pd.DataFrame(results, columns=['Vectorizer',
'Classifier', 'Model', 'Dataset', 'Accuracy'])

# Compute average accuracy by Dataset and Vectorizer
average_results = results_df.groupby(['Dataset',
'Vectorizer'])['Accuracy'].mean().reset_index()

# Print the average results table
print("Average Results:")
print(average_results)

# Visualize average accuracy by Dataset and Vectorizer
fig, ax = plt.subplots(figsize=(10, 6))
for dataset in average_results['Dataset'].unique():
    dataset_results =
average_results[average_results['Dataset'] == dataset]
    ax.plot(dataset_results['Vectorizer'],
dataset_results['Accuracy'], marker='o', label=dataset)

ax.set_title('Average Accuracy by Dataset and Vectorizer')
ax.set_xlabel('Vectorizer')
ax.set_ylabel('Accuracy')
ax.legend()
plt.xticks(rotation=45)
plt.show()


import pandas as pd
import matplotlib.pyplot as plt

# Convert the results to a DataFrame
results_df = pd.DataFrame(results, columns=['Vectorizer',
'Classifier', 'Model', 'Dataset', 'Accuracy'])
```

```
# Compute average accuracy by Dataset and Classifier
average_results = results_df.groupby(['Dataset',
'Classifier'])['Accuracy'].mean().reset_index()

# Print the average results table
print("Average Results:")
print(average_results)

# Visualize average accuracy by Dataset and Classifier
fig, ax = plt.subplots(figsize=(10, 6))
for dataset in average_results['Dataset'].unique():
    dataset_results =
average_results[average_results['Dataset'] == dataset]
    ax.plot(dataset_results['Classifier'],
dataset_results['Accuracy'], marker='o', label=dataset)

ax.set_title('Average Accuracy by Dataset and Classifier')
ax.set_xlabel('Classifier')
ax.set_ylabel('Accuracy')
ax.legend()
plt.xticks(rotation=45)
plt.show()

!pip install plotly


!pip install nbformat


pip install ipykernel


pip install --upgrade nbformat
pip install transformers

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier,
LogisticRegression, SGDClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import LabelEncoder

# Convert string labels to binary integers
label_encoder = LabelEncoder()
```

```
df1['label'] = label_encoder.fit_transform(df1['label'])
df4['label'] = label_encoder.fit_transform(df4['label'])
df5['deception'] =
label_encoder.fit_transform(df5['deception'])
df2['label'] = label_encoder.fit_transform(df2['label'])

# Define your datasets
datasets = [
    (df1['text'], df1['label'], 'Fake news during normal
times - 1'),
    (df4['article_content'], df4['label'], 'Syrian war fake
news'),
    (df5['content'], df5['deception'], 'Iraq war fake news'),
    (df2['text'], df2['label'], 'Fake news during normal
times - 2')
]

# Define your classifiers and vectorizers
classifiers = [PassiveAggressiveClassifier(),
LogisticRegression(), SGDClassifier()]
vectorizers = [TfidfVectorizer(),
CountVectorizer(),HashingVectorizer()]

# Define the evaluation metrics you want to use
metrics = {
    'Accuracy': accuracy_score,
    'Precision': precision_score,
    'Recall': recall_score,
    'F1 Score': f1_score,
    'ROC AUC': roc_auc_score
}

# Perform train-test split and evaluate models
results = []

for dataset in datasets:
    X, y, dataset_name = dataset
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=7)

    for classifier in classifiers:
        for vectorizer in vectorizers:
            pipeline = make_pipeline(vectorizer, classifier)
            pipeline.fit(x_train, y_train)
            y_pred = pipeline.predict(x_test)

            evaluation_scores = {}
            for metric_name, metric_func in metrics.items():
                if metric_name == 'Precision':
                    score = metric_func(y_test, y_pred,
pos_label=1)
```

```
                else:
                    score = metric_func(y_test, y_pred)
                evaluation_scores[metric_name] = score

            results.append({
                'Dataset': dataset_name,
                'Classifier': classifier.__class__.__name__,
                'Vectorizer': vectorizer.__class__.__name__,
                **evaluation_scores
            })

# Convert the results to a DataFrame
results_df = pd.DataFrame(results)

# Print the results
print(results_df)

results_df.head(200)
print(results_df.to_string())

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer,
HashingVectorizer, CountVectorizer
from sklearn.linear_model import PassiveAggressiveClassifier,
LogisticRegression, SGDClassifier
from sklearn.metrics import accuracy_score


# Convert string labels to binary integers
label_encoder = LabelEncoder()
df1['label'] = label_encoder.fit_transform(df1['label'])
df4['label'] = label_encoder.fit_transform(df4['label'])
df5['deception'] =
label_encoder.fit_transform(df5['deception'])
df2['label'] = label_encoder.fit_transform(df2['label'])


# Define your datasets
datasets = [
    (df1['text'], df1['label'], 'Fake news during normal
times - 1'),
    (df4['article_content'], df4['label'], 'Syrian war fake
news'),
    (df5['content'], df5['deception'], 'Iraq war fake news'),
    (df2['text'], df2['label'], 'Fake news during normal
times - 2')
]


# Define your classifiers and vectorizers
classifiers = [PassiveAggressiveClassifier(),
LogisticRegression(), SGDClassifier()]
vectorizers = [TfidfVectorizer(), CountVectorizer(),
HashingVectorizer()]


# Perform train-test split and evaluate models
results = []


for dataset in datasets:
    X, y, dataset_name = dataset
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=7)


    for classifier in classifiers:
        for vectorizer in vectorizers:
            pipeline = make_pipeline(vectorizer, classifier)
            pipeline.fit(x_train, y_train)
            y_train_pred = pipeline.predict(x_train)
            train_accuracy = accuracy_score(y_train,
y_train_pred)


            results.append({
                'Dataset': dataset_name,
                'Classifier': classifier.__class__.__name__,
                'Vectorizer': vectorizer.__class__.__name__,
                'Train Accuracy': train_accuracy
            })


# Convert the results to a DataFrame
results_df = pd.DataFrame(results)


# Print the results
print(results_df)
```