Fake News Detection Using Machine Learning

UNIVERSITY OF
LINCOLN

Khadija Altaf
27739598

27739598@students.lincoln.ac.uk

School of Computer Science

College of Science

University of Lincoln

Submitted in partial fulfilment of the requirements for the
Degree of MSc in Data Science and Applied Analytics

Supervisor: Dr Leonardo Guevara

September 2024

Acknowledgements

I want to thank my project supervisor, Dr Leonardo Guevara, for his continuous help and support throughout this project, which helped with project direction. Dr Guevara also helped me find a balanced dataset to use for this project which I am extremely grateful for.

I want to express my sincere gratitude to the professors and lecturers in the School of Computer Science at the University of Lincoln. Their teachings and advice have given me a strong foundation for my research project, and I am genuinely thankful for the knowledge and skills they have shared with me.

I would also like to acknowledge the invaluable resources provided by the University of Lincoln, which enabled me to do this research project. The access to literature, computational tools, and academic resources was crucial to the success of this project.

I am deeply grateful to my parents and siblings for their support, patience, and understanding throughout this process. Their encouragement has been a constant source of motivation, especially during the more challenging phases of my work.

Abstract

This project utilises the LIAR dataset to demonstrate the use of machine learning methods in categorising political statements based on their truthfulness. In today's environment, where misinformation is increasingly common, the automatic assessment of the accuracy of public statements is essential. The study concentrates on the development and comparison of various classification models, such as Logistic Regression, Support Vector Machines (SVM), and Random Forests, along with feature extraction techniques like Term Frequency-Inverse Document Frequency (TF-IDF), Bag of Words, Word2Vec, and GloVe.

The dataset goes through thorough preprocessing, which includes tokenisation, lemmatisation, and noise removal. Then, feature extraction is done to convert the text data into numerical representations. I use TF-IDF and Bag of Words to measure the importance of words across the dataset, and Word2Vec and GloVe to create semantic word embeddings.

The models are trained and evaluated using these features, with performance metrics such as accuracy, precision, recall, and F1-score being analysed. Despite applying sophisticated techniques, initial results show moderate performance, particularly distinguishing between similar truthfulness categories. To address this, the study further refines the classification by simplifying the labels into binary classes—'True' and 'Fake'—significantly improving the model's accuracy.

The findings indicate that although machine learning models can offer useful insights into the truthfulness of statements, the intricacy and subtlety of language pose significant challenges. This study adds to the continuous work in automating fact-

checking and emphasises areas for potential improvement, especially in feature representation and model tuning.

Table of Contents

Chapter 1

Introduction

In today's rapidly evolving digital landscape, the spread of misinformation has emerged as a significant global challenge. The proliferation of false or misleading information across social media platforms and other online channels has made it increasingly difficult for individuals to discern fact from fiction. Traditional fact-checking methods, which rely on experts manually verifying information, are becoming increasingly inadequate given the sheer volume of data generated and shared daily. This inability to keep pace with the spread of information underscores the urgent need for more scalable and efficient solutions to address the problem of misinformation. (ACM Conferences, 2022)

Recent advancements in machine learning (ML) and Natural Language Processing (NLP) have opened new avenues for automating the fact-checking process. These technologies hold the potential to revolutionise how we detect and classify misinformation by creating models capable of automatically assessing the truthfulness of statements. By leveraging machine learning and NLP, it is possible to significantly enhance the speed and accuracy of fact-checking efforts, thereby mitigating the impact of false information on public discourse. This research project aims to contribute to this field by exploring and comparing different machine learning models and feature extraction techniques for classifying the truthfulness of political statements.

The study will utilize the LIAR dataset, introduced by Wang, 2017, a well-established benchmark in the realm of automated fact-checking, which comprises thousands of labelled statements sourced from Politifact. The dataset's labels—ranging from "true" and "mostly true" to "half-true," "barely true," "false," and "pants on fire"—capture the nuanced spectrum of truthfulness in political discourse. These labels present a unique challenge in machine learning, as they

require models to not only identify outright falsehoods but also distinguish between varying degrees of truth. This research seeks to determine whether machine learning models can effectively navigate this complexity and whether simplifying the labels into binary classes (e.g., true vs. fake) can enhance the accuracy and reliability of the classification process.

1.1 Aims

The primary aim of this research is to develop and rigorously evaluate machine learning models for detecting fake news using the LIAR dataset. This objective involves a thorough exploration of various text preprocessing techniques, feature extraction methods, and machine learning algorithms to identify the most effective strategies for accurately classifying fake news. By focusing on the LIAR dataset, this research intends to contribute to the academic and practical understanding of fake news detection, offering potential solutions that can be applied to real-world scenarios.

1.2 Objectives

To conduct a comprehensive literature review on existing machine learning models and NLP techniques for fake news detection: This objective involves reviewing current methodologies and identifying the most effective approaches for detecting fake news. The literature review will provide a theoretical foundation for the research and highlight gaps that this study aims to address.

To develop and compare various feature extraction techniques for transforming textual data: The study will examine and evaluate different methods for converting text into numerical representations suitable for machine learning. Techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) and word embeddings (e.g., Word2Vec, GloVe) will be explored to determine which provides the most informative features for the classification task.

To develop, train, and evaluate both traditional and advanced machine learning models for fake news detection: The research will involve training

models such as Logistic Regression, Support Vector Machines (SVM), and Random Forest on preprocessed data. The performance of these models will be assessed to identify the most effective for fake news classification. If time permits, the study will also extend to advanced models, including neural networks and transformers, with efforts focused on optimising their performance through hyperparameter tuning and other techniques.

Through these objectives, this research seeks to advance the field of automated fact-checking and contribute valuable insights into the development of robust machine-learning models for detecting fake news.

Chapter 2

Literature Review

2.1 Impact of Fake News on Society

The emergence of social media has transformed the way news is created and shared, changing traditional media into digital platforms. This shift has allowed more people to participate in news sharing, but it has also led to the rapid spread of false information. Platforms like Facebook, Twitter, and YouTube have become major sources of news and misinformation, shaping public opinion and political processes (Columbia.edu, 2023). Furthermore, the direct interaction of political leaders with the public through these channels has further increased the impact of false news on society. (Olan et al., 2022)

2.2 Need for Automated Fake News Detection

Traditional fact-checking methods have historically relied on journalists and dedicated fact-checkers manually verifying information. This process requires cross-referencing claims with credible sources, consulting experts, and thoroughly reviewing documents or evidence to establish the accuracy of a statement (Team, 2023). While this approach is highly reliable and ensures that only verified information is shared with the public, it has significant limitations. One of the primary challenges is scalability—manual fact-checking is time-consuming and resource-intensive, making it difficult to keep up with the rapid spread of information in today's digital age. Additionally, the speed at which news is produced and shared, particularly on social media platforms (Manish Kumar Singh et al., 2023), often outpaces the ability of fact-checkers to verify claims before they reach a wide audience. As a result, false information can spread quickly and extensively, causing considerable harm before it can be corrected.

These limitations emphasize the need for more efficient and scalable solutions in the era of fast-paced information exchange.

The reliability of internet information has become a significant concern, especially with the increasing spread of misinformation and fake news. Studies have shown that fake news can significantly influence political and social realities, as seen during events such as the 2016 US presidential elections (State, 2016) and the 2013 Boston Marathon bombings (Schultz, 2013). Traditional methods of verifying online content, such as manual fact-checking, are becoming increasingly impractical due to the sheer volume of information and the speed at which it spreads on social media platforms. This has led to a growing need for automated fake news detection systems that can efficiently handle the rapid dissemination of false information (Della et al., 2018).

Automated fake news detection methods are generally categorised into content-based and social-based approaches. Content-based methods analyse the textual content of news articles, while social-based methods use social signals, such as user engagement on platforms like Twitter and Facebook, to identify fake news. Each approach has its strengths and limitations, but combining them can enhance detection accuracy, especially in situations where social signals are scarce or unreliable. This hybrid approach has been shown to outperform traditional methods, making it a promising solution for early detection and prevention of fake news in real-world applications. (Della et al., 2018)

2.3 Natural Language Processing

Natural Language Processing (NLP) is a key area in the field of artificial intelligence (AI) that aims to enable machines to understand, interpret, and generate human language in a meaningful and useful manner (GeeksforGeeks, 2021). This field combines computer science, linguistics, and AI to bridge the communication gap between humans and computers, allowing for more natural and efficient interactions. NLP is particularly important in its ability to process

and analyse large amounts of unstructured text data – data that does not follow a predetermined format or structure. As digital text data continues to grow exponentially, NLP has become essential for extracting meaningful insights, automating language-related tasks, and enabling machines to perform tasks that traditionally require human language comprehension.

The applications of NLP are extensive and varied, spanning across multiple domains (Ibm.com, 2021). For instance, sentiment analysis is a widely used NLP application that involves determining the emotional tone or sentiment behind a piece of text. This technique is especially useful in marketing and customer service, where understanding customer sentiment can guide business strategies and improve customer experiences. Another critical application is topic modelling, which helps in identifying the underlying themes or topics within a large corpus of text. This technique is beneficial for summarising vast amounts of information and gaining insights into trends within large datasets. Additionally, text classification, an NLP task where text data is categorised into predefined classes, is fundamental in spam detection, content moderation, and even legal document classification. These NLP-driven applications have revolutionised industries such as finance, healthcare, and marketing by enabling more efficient and data-driven decision-making processes.

NLP is a crucial part of this research, which focuses on using machine learning techniques to detect fake news. In today's digital age, social media platforms and news outlets produce a huge amount of textual data every day. Traditional methods of analysing and verifying this information have proven to be insufficient due to the large volumes of data. NLP provides the necessary tools to automatically process and analyse this text data, allowing for the detection of patterns, linguistic features, and inconsistencies often associated with fake news. This research aims to develop a more accurate and scalable solution for detecting fake news by leveraging NLP techniques such as text classification and word embeddings. This is a critical challenge with significant implications for the integrity of information in society. The ability of NLP to handle unstructured text data and extract relevant

features underpins the effectiveness of the models developed for this project. These models are designed to distinguish between true and false information with greater accuracy. This approach not only addresses a pressing issue in today's information-driven world but also contributes to the broader field of NLP by exploring its applications in a critical area of societal impact.

2.4 Machine Learning for Text Classification

Machine learning is now an essential aspect of text classification, which involves organising text data into pre-established classes or categories. This is especially important in automated fact-checking, where the objective is to categorise text as true or false, commonly known as fake news detection. Text classification with machine learning techniques utilises the extensive amount of digital text data accessible today, offering an effective and scalable solution for processing and analysing this information. (Kamran Kowsari et al., 2019)

In text classification, machine learning models are trained on labelled datasets where the categories are predefined, such as 'True' or 'Fake' in the case of fact-checking. The models learn to recognise patterns, linguistic features, and other relevant indicators within the text data that correlate with these categories. Once trained, these models can predict the category of new, unseen text data, making them powerful tools for automated text analysis.

Several machine learning algorithms are commonly used for text classification tasks (Zhang and Oles, 2001):

1. **Logistic Regression:** Logistic regression is one of the most commonly used algorithms for binary classification tasks. In text classification, it estimates the probability that a given piece of text belongs to a particular class based on features extracted from the text, such as word frequencies or word embeddings (GeeksforGeeks, 2024). Despite its simplicity,

logistic regression is highly effective and often used as a baseline model for more complex algorithms.

2. **Support Vector Machines (SVM):** Support Vector Machine (SVM) is a powerful machine learning algorithm that works well in high-dimensional spaces, making it ideal for text classification with a large number of features, such as words or n-grams (scikit-learn, 2024). SVM operates by identifying the hyperplane that best separates the data into distinct classes with the maximum margin. Known for its capability to handle intricate decision boundaries, SVM is widely used in text classification tasks, particularly in scenarios involving subtle differences between categories.
3. **Random Forests:** The Random Forest method is an ensemble learning technique that creates many decision trees during training and then gives the most commonly occurring class (for classification) or the average prediction (for regression) from the individual trees (Shafi, 2018). In text classification, Random Forests are especially beneficial because they can handle large datasets with high dimensionality and are resistant to overfitting. By combining the results from multiple decision trees, Random Forests can generate more reliable and accurate predictions.

These algorithms have been widely used in various text classification tasks beyond fact-checking, such as sentiment analysis, spam detection, and topic categorisation. In the context of fake news detection, machine learning models are trained on datasets containing both truthful and false information, learning to identify the characteristics that set them apart. The choice of algorithm often depends on specific task requirements, such as the dataset size, text data nature, and the desired balance between accuracy and computational efficiency.

In summary, machine learning for text classification is a critical tool in automating fact-checking processes. By using algorithms such as Logistic Regression, Support Vector Machines, and Random Forests, models can be created that

efficiently and accurately classify text data. This contributes to the broader effort to combat the spread of misinformation.

2.5 Feature Extraction Techniques

2.5.1 TF-IDF(Term Frequency-Inverse Document Frequency)

TF-IDF is a popular technique in Natural Language Processing (NLP) used to convert textual data into numerical features. This method helps to measure the significance of words within a document when compared to a collection of documents (corpus). TF-IDF is especially useful in text classification tasks, such as spam detection, sentiment analysis, and fake news detection, as it allows for differentiation between commonly used words and those that are more unique or informative within specific documents (Zilliz.com, 2024).

The TF-IDF score for each term in a document is calculated as the product of two components: Term Frequency (TF) and Inverse Document Frequency (IDF) (GeeksforGeeks, 2021). Term Frequency refers to the number of times a word appears in a document, normalised by the total number of words in that document. It reflects how important a word is within a particular document. However, some words might frequently appear in many documents across the corpus, diluting their significance. This is where Inverse Document Frequency comes into play.

The IDF measures the rarity of a word across a corpus by taking the logarithm of the total number of documents divided by the number of documents containing that word. A high IDF score indicates that the word is rare across the corpus, while a low score suggests that the word is common. By multiplying TF (Term Frequency) by IDF, TF-IDF gives higher scores to words that are frequent in a document but rare across the corpus, making them more valuable for distinguishing that document from others (GeeksforGeeks, 2021).

In the realm of fake news detection, TF-IDF plays a crucial role in identifying and highlighting the distinct language patterns or terms that might be more prominent

in deceptive or untrue articles as opposed to genuine ones. By transforming text into these weighted numerical characteristics, machine learning models can gain a deeper comprehension of the data and consequently make more precise and dependable predictions. This approach's simplicity and efficacy make it an essential tool in the preprocessing pipeline for numerous NLP tasks.

2.5.2 Bag of Words

The CountVectorizer is an important tool in natural language processing (NLP) that is used to convert text data into numerical features (Kumar, 2024). This technique, also known as the "Bag of Words" model, transforms a set of text documents into a matrix of token counts. Each unique word in the corpus is considered a feature, and its corresponding value reflects the number of times the word appears in the document. The result is a sparse matrix where rows represent documents and columns represent words, with the entries reflecting word frequencies. Despite its simplicity, the CountVectorizer is powerful for tasks such as text classification, sentiment analysis, and topic modelling, as it captures the basic structure of the text data without considering the order of words, focusing only on the occurrence of terms. However, one limitation of the Bag of Words model is that it does not capture context or semantic meaning, treating words as independent entities. This method often serves as a starting point for more advanced NLP techniques, such as TF-IDF or word embeddings, which aim to address some of these limitations by incorporating more information about the context and significance of words within documents.

2.5.3 Word Embedding

Word embeddings have transformed the field of Natural Language Processing (NLP) by allowing machines to comprehend and analyse human language more effectively. These embeddings consist of dense vector representations of words, capturing semantic relationships and contextual similarities between them (TensorFlow, 2022). Two of the most notable word embedding techniques are Word2Vec and GloVe.

Word2Vec

Word2Vec, developed by a team led by Tomas Mikolov at Google in 2013, is one of the earliest and most influential word embedding techniques. It represents words as continuous vector spaces, where similar words have similar vector representations. The model is based on the distributional hypothesis, which suggests that words appearing in similar contexts tend to have similar meanings (Alammar, 2015). Word2Vec operates through two main architectures:

Continuous Bag of Words (CBOW): This approach predicts a target word based on the words surrounding it in context. For example, if the sentence is "The cat sits on the __," CBOW aims to predict the word "mat." CBOW is computationally efficient and performs well with large datasets, making it suitable for training on extensive corpora. (Alammar, 2015)

Skip-Gram: In contrast, the Skip-Gram model works by predicting the surrounding context words when given a target word. For example, if the target word is "cat," Skip-Gram will predict the context words "the," "sits," and "on." This model is particularly effective with smaller datasets and it is very good at capturing complex semantic relationships between words. (Alammar, 2015)

GloVe

GloVe (Global Vectors for Word Representation) is a widely used word embedding technique developed by researchers at Stanford University in 2014 (GeeksforGeeks, 2022). It shares similarities with Word2Vec but differs in its approach to learning word representations.

GloVe combines the strengths of matrix factorisation methods such as Latent Semantic Analysis (LSA) with the context-based learning approach of Word2Vec. It creates a global word co-occurrence matrix where each element indicates how often two words appear together in a specific context across the entire corpus

(GeeksforGeeks, 2022). GloVe then decomposes this matrix to create word vectors, capturing both global statistical information and local context.

Word2Vec and GloVe are both powerful tools for word embedding, but they serve different purposes. Word2Vec is great at capturing local context and semantic relationships, making it suitable for tasks like analogy solving and semantic analysis (GeeksforGeeks, 2018). On the other hand, GloVe excels at capturing global context, making it more effective for tasks that require understanding the overall distribution of words across a corpus (GeeksforGeeks, 2022).

In the context of researching fake news detection, word embeddings such as Word2Vec and GloVe play a crucial role in converting textual data into a format that can be processed by machine learning models. These embeddings capture the subtle meanings of language, enabling the identification of patterns that could indicate the truthfulness of a statement. Word2Vec's ability to capture semantic relationships and GloVe's understanding of global context both improve the accuracy of fake news detection models, making them invaluable tools in addressing the challenges of misinformation.

Comparison between Word2Vec and GloVe

Word2Vec and GloVe are two prominent word embedding techniques. Although they have similar purposes, they differ significantly in their approaches (Verma, 2021). Word2Vec, developed by Google, focuses on capturing the local context of words through either the Continuous Bag of Words (CBOW) or Skip-Gram models. It excels in representing semantic relationships between words, making it ideal for tasks like analogy solving. However, it tends to ignore the global context of the corpus, potentially missing broader relationships.

On the other hand, GloVe, developed by Stanford, captures both local and global contexts by utilising a co-occurrence matrix, which records the frequency of word pairs appearing together. This enables GloVe to offer a more thorough

comprehension of word relationships throughout a corpus. It proves especially effective in tasks that demand an understanding of global context, such as information retrieval and language modelling.

In the context of fake news detection, both techniques have their advantages. Word2Vec's emphasis on local context assists in spotting nuanced word usage, whereas GloVe's global perspective aids in comprehending broader patterns. Integrating insights from both methods could result in a more reliable and precise model for identifying misinformation.

2.6 Automated Fact-Checking: Current Approaches and Models

The effort to combat fake news has resulted in significant progress in developing fact-checking models, primarily through the use of machine learning (ML) and natural language processing (NLP) techniques. These models aim to effectively differentiate between real and fake news, addressing one of the most urgent issues in today's information-driven society. The literature presents several notable approaches that employ different methods, feature extraction techniques, and machine learning models, all contributing to the increasing knowledge in this field.

2.6.1 Political News and ML Pipelines

A recent study by Agarwal, Sandeep Reddivari and Kalyan Reddivari (2022) focused on the issue of fake news within the realm of political news. The research involved creating a pipeline that utilised feature extraction methods such as Count Vectorizer and TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. These methods are essential in Natural Language Processing (NLP) for converting textual data into numerical representations suitable for machine learning models. The study utilised five different machine learning models: Support Vector Machine (SVM), Random Forest, Logistic Regression, Decision Tree, and Multinomial Naive Bayes. These models were trained and tested on the LIAR dataset, a commonly used dataset for fake news detection, which contains statements labelled

across six categories. For this study, the labels were simplified into a binary classification system, where categories like "pants-fire," "false," and "barely-true" were grouped as 'Fake,' while "half-true," "mostly-true," and "true" were categorised as 'Real.'

The preprocessing phase was a critical aspect of the research, emphasising the significance of data quality for accurate predictions. The original LIAR dataset contained misaligned text, garbage values, and unnecessary symbols, which could potentially lead to inaccurate predictions if not properly addressed. The researchers meticulously aligned the text manually, removed irrelevant content such as IP addresses, URLs, and stop words, and applied stemming to ensure that the text was in its most useful form. Subsequently, the dataset was divided into 80% training data and 20% testing data to effectively train the models.

The study found that different machine learning (ML) models had varying performances depending on the method used for feature extraction. Logistic Regression (LR) was the top performer with an average accuracy of 61% when combined with the TF-IDF vectorizer. On the other hand, Multinomial Naive Bayes (MNB) achieved a 60% accuracy using the Count Vectorizer method. To improve the model's performance, the researchers used 5-fold cross-validation and Grid Search Cross-validation to fine-tune the model parameters. These steps highlight the importance of model selection and hyperparameter tuning in developing effective fake news detection systems.

2.6.2 Diverse NLP Text Vectorization and ML Classifiers

In a study conducted by Smitha Nl and Bharath R in 2020, a significant contribution was made to the field. The study aimed to detect fake news using various NLP text vectorization techniques with different ML classifiers. The researchers obtained their dataset from Kaggle, which included approximately 13,000 posts collected from 244 websites over 30 days. This extensive dataset provided a solid foundation for experimenting with different methodologies.

The preprocessing steps in this study were comprehensive and rigorous, ensuring that the text data was cleaned and ready for analysis. All letters in the documents were converted to lowercase, numbers were removed, and punctuation and accent marks were stripped away. The researchers also eliminated unnecessary white spaces and stop words, which are common practices in text preprocessing to enhance the quality of the input data.

The study examined various methods for vectorizing text, including Count Vectorizer and TF-IDF, as well as word embeddings. These techniques were used to train different machine learning models, such as SVM, Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and XG-Boost, to assess their performance in classifying news articles as fake or real. The researchers found that the effectiveness of the models varied significantly depending on the vectorization technique. For example, a neural network model outperformed other classifiers with an impressive accuracy of 94% when using the Count Vectorizer method. Similarly, SVM combined with TF-IDF vectorizer also achieved a 94% accuracy, indicating its effectiveness in this context. The study also showed that when word embeddings were used, the neural network still performed well, achieving a 90% accuracy, confirming the robustness of neural networks in fake news detection tasks.

The knowledge acquired from these studies is crucial to my research. I am working on creating a dependable and adaptable system for detecting fake news using machine-learning techniques. By utilising NLP methods such as TF-IDF and word embeddings alongside different ML models, I aim to establish a robust understanding of the capabilities and constraints of various approaches.

The first study's focus on political news and the meticulous data preprocessing steps offer valuable lessons for my work. For example, the importance of ensuring data quality through processes like manual alignment and removal of irrelevant content is critical for achieving accurate model predictions. Additionally, the use of binary classification, as seen in the transformation of the LIAR dataset, aligns with my

research goal of simplifying the detection process to enhance the model's efficiency and effectiveness.

Furthermore, the second study's investigation of various vectorization methods and machine learning classifiers provides a chance to assess different combinations that may enhance the accuracy of my models. The discovery that neural networks and SVM performed exceptionally well with specific vectorization techniques implies that these models could be promising candidates for my research. By employing a similar approach, I intend to explore different NLP and ML techniques to determine the most effective strategy for identifying fake news.

Ultimately, these studies provide a comprehensive understanding of the current state of fake news detection and offer practical guidance for advancing my research. By leveraging the knowledge and methodologies outlined in these works, I can develop a more accurate and scalable solution to address the critical issue of fake news in today’s digital age.

2.7 Evaluation Metrics in Text Classification

Evaluation metrics are essential for assessing the performance of text classification models, particularly in tasks such as fake news detection. The commonly used metrics include accuracy, precision, recall, and F1-score, each offering a different perspective on the model's effectiveness.

Accuracy is a simple metric that measures the proportion of correctly classified instances out of the total instances (GeeksforGeeks, 2020). While accuracy provides an overall view of model performance, it can be deceptive when dealing with imbalanced datasets. In instances where one class dominates, accuracy may be high, but the model's performance in identifying the minority class could be poor.

Precision measures the model's accuracy in identifying positive instances, calculated as the ratio of true positives to the sum of true positives and false positives (GeeksforGeeks, 2020). In the context of fake news detection, precision addresses the question: "Out of all the articles classified as fake, how many are actually fake?" High precision is essential in situations where incorrectly labelling real news as fake (false positives) can have serious consequences, like eroding trust in reliable news sources.

Recall, also known as sensitivity, measures the model's ability to capture all relevant instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives (GeeksforGeeks, 2020). In the context of fake news detection, recall answers the question: "Of all the fake news articles, how many did the model correctly identify?" High recall is important when the cost of missing a fake news article is high because undetected fake news can spread misinformation widely.

F1-score is calculated as the harmonic mean of precision and recall, which helps to strike a balance between the two (GeeksforGeeks, 2020). It is especially valuable when there is a requirement to balance precision and recall, ensuring that the model excels at identifying true positives and minimizing false negatives.

In the field of fact-checking, these measurements are important for assessing how well a model can differentiate between true and false statements. While high accuracy is important, it's equally crucial to have strong precision and recall. Without these, a model may struggle to accurately identify fake news or may mistakenly flag real news, leading to unintended consequences. As a result, a balanced F1-score is often the most informative indicator of a model's effectiveness in fact-checking, as it provides a comprehensive assessment of the model's performance in various situations.

Confusion Matrix: In text classification, evaluating the performance of a model is crucial to understanding how well it distinguishes between different classes. One

of the most widely used evaluation tools for this purpose is the confusion matrix. The confusion matrix is a table that provides a comprehensive snapshot of the classification results by displaying the true positives, true negatives, false positives, and false negatives (GeeksforGeeks, 2020). These metrics are arranged in a matrix format, where the rows typically represent the actual classes, and the columns represent the predicted classes.

The confusion matrix is extremely useful because it allows us to not only assess the overall accuracy of a model but also to see how well it performs in each individual class. In a binary classification problem, a model's accuracy might be high, but if the dataset is imbalanced (one class is much more frequent than the other), the model might be biased towards predicting the majority class. The confusion matrix reveals this by showing how often the model confuses one class for the other.

From the confusion matrix, several other important metrics can be derived, including **precision**, **recall**, **F1-score**, and **specificity**. Precision measures the proportion of positive identifications that were actually correct, while recall (also known as sensitivity) measures the proportion of actual positives that were correctly identified. The F1-score is the harmonic mean of precision and recall, providing a single metric that balances the two, particularly useful when dealing with imbalanced datasets. Specificity, on the other hand, measures the proportion of actual negatives that were correctly identified.

2.8 Gaps and Conclusion

Despite extensive research on detecting fake news using machine learning techniques, there are still several gaps that require further investigation. One major gap is the reliance on traditional feature extraction methods such as Count Vectorizer and TF-IDF. While these techniques have been effective in converting text data into numerical formats for model training, they often fail to capture the complex, nuanced patterns present in fake news. These methods primarily focus on

word frequency and overlook the contextual and semantic relationships between words, which are crucial for understanding the underlying meaning of a statement.

Another gap is the limited exploration of advanced NLP techniques, such as word embeddings and transformer-based models like BERT, in the context of fake news detection. Although these methods have shown promise in other text classification tasks by capturing deep semantic meanings and context, their application in fake news detection remains underexplored. Additionally, existing studies often do not adequately address the challenges posed by the dynamic and evolving nature of fake news, where the language and tactics used to spread misinformation constantly change.

Furthermore, much of the research has focused on binary classification, where news is labelled as either true or fake. However, this oversimplifies the complexity of misinformation. This binary approach fails to consider the varying degrees of truthfulness in statements, which can range from completely false to partially true. For example, the LIAR dataset categorizes statements into six levels of truthfulness, but when these categories are transformed into a binary format, important information is lost, which could potentially improve detection accuracy.

This research aims to explore the use of advanced NLP techniques, such as word embeddings, to improve the accuracy and robustness of fake news detection. By focusing on the nuanced differences in truthfulness, this study seeks to develop a more comprehensive and adaptable model that better reflects the complexities of real-world misinformation.

The literature review has highlighted the significant progress made in detecting fake news using machine learning techniques, particularly through traditional methods such as Count Vectorizer and TF-IDF. These techniques have been widely used, showing varying degrees of success across different machine-learning models. However, the review also identifies several gaps, including the limitations of these traditional methods in capturing language's semantic and contextual

nuances, the underutilisation of advanced NLP techniques such as word embeddings and transformers, and the oversimplification of fake news detection into binary classifications.

In the following chapter, I will outline the methodology used in this research. This approach aims to create a more efficient model for detecting fake news, taking into account the intricacies of misinformation in today's digital environment.

Chapter 3

Methodology

3.1 Research Design

This study uses a quantitative approach to examine how statements can be classified based on their truthfulness using the LIAR dataset. The main goal is to create and evaluate models that can differentiate between honest and deceptive statements by using natural language processing (NLP) techniques and machine learning algorithms. The methodology is organised into stages: data preprocessing, feature extraction, and model training and evaluation ensuring that the analysis is thorough and replicable.

The research originally aimed to categorise statements based on the original six labels provided in the LIAR dataset: "true," "mostly true," "half true," "barely true," "false," and "pants on fire." However, due to the complexity and subjectivity of these nuanced labels, the models had difficulty performing well, especially in terms of accuracy. It was challenging for the models to accurately differentiate between these closely related categories, resulting in unsatisfactory predictive power.

Given these challenges, the research shifted to a binary classification approach. The six original labels were combined into two broader categories: "True" (which includes "true," "mostly-true," and "half-true") and "Fake" (comprising "barely-true," "false," and "pants-on-fire"). This simplification aimed to improve the models' performance by enhancing classification accuracy. The binary approach also aligns with the primary goal of distinguishing truth from falsehood, making it a more practical and understandable framework for the research.

3.2 Data Collection

3.2.1 Dataset Overview

The data set used in this research is the LIAR data set, which is a comprehensive resource for analysing the truthfulness of public statements. Created by Wang (2017), the data set contains 12,836 statements that have been fact-checked and labelled by PolitiFact.com into six categories based on their veracity, making it an ideal candidate for supervised machine learning tasks. These labels provide a nuanced view of the truthfulness of each statement, ranging from completely true to blatantly false.

In addition to the truthfulness labels, each statement in the LIAR dataset is accompanied by metadata that includes the speaker's name, job title, political affiliation, the subject of the statement, and contextual information. This metadata is valuable for exploring potential correlations between the speaker's background and the truthfulness of their statements, contributing to a deeper understanding of the factors influencing truthfulness.

The dataset is divided into three subsets: a training set with 10,269 statements, a validation set with 1,284 statements, and a test set with 1,283 statements. These subsets are used for training, validating, and testing the models. This ensures that the results can be applied to new, unseen data in a generalisable way.

The dataset’s structure supports the application of various statistical and machine learning methods, facilitating the extraction of measurable outcomes such as accuracy rates. By relying on this dataset, the research benefits from a well-established and widely recognised source of labelled data, which enhances the credibility and reliability of the findings.

3.2.2 Data Preprocessing

Data preprocessing is an essential step in preparing the dataset for analysis. The raw text data must be cleaned and transformed into a format suitable for machine learning models. The preprocessing steps included the following:

Loading the Data: The LIAR dataset was loaded from TSV (Tab-Separated Values) files using the Pandas library in Python. The training, validation, and test sets were combined into a single DataFrame to facilitate preprocessing. The data was shuffled to ensure randomness in the distribution of statements.

```
import pandas as pd

# Importing the dataset
train = pd.read_csv("C:\\Users\\KHADIJA ALTAF\\Downloads\\liar_dataset\\train.tsv", sep='\t', quoting=3, header=None)
test = pd.read_csv("C:\\Users\\KHADIJA ALTAF\\Downloads\\liar_dataset\\test.tsv", sep='\t', quoting=3, header=None)
valid = pd.read_csv("C:\\Users\\KHADIJA ALTAF\\Downloads\\liar_dataset\\valid.tsv", sep='\t', quoting=3, header=None)

df_raw = pd.concat([train, test, valid], axis=0, sort=False)
df_raw = df_raw.sample(frac=1).reset_index()
```

Figure 1 Loading the Data

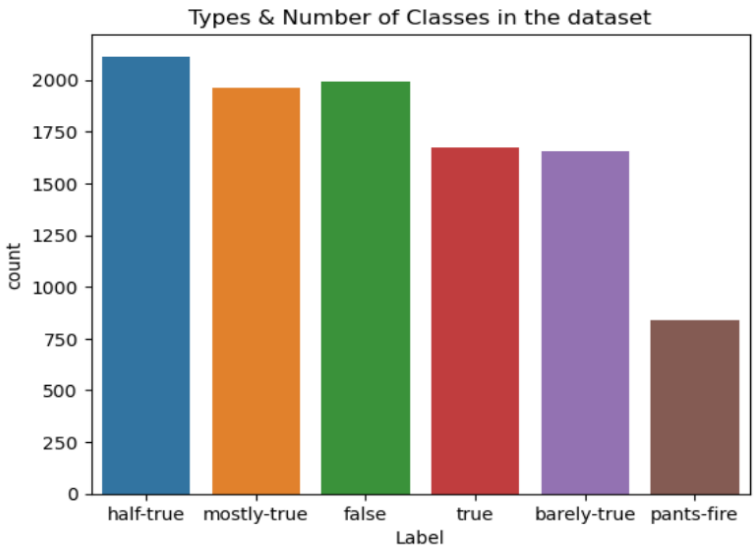

Figure 2 Types & Number of Classes in the Dataset

Text Cleaning: The raw text data often contains noise, such as HTML tags, punctuation, and stop words, which do not contribute to the meaning of the text. To clean the data, the following steps were taken:

HTML Tags Removal: HTML tags were removed from the text using the BeautifulSoup library. Removing these tags is essential to eliminate unnecessary noise from the data.

Punctuation Removal: Punctuation marks were removed using the string library in Python. This step is crucial for reducing the dimensionality of the data, as punctuation typically does not contribute to the semantic meaning of the text.

Stop Words Removal: Stop words, which are common words like "the," "is," "in," etc., were removed using the NLTK library. These words generally do not add significant meaning to the text and can be excluded to focus on more meaningful terms.

```
import re
import string
from bs4 import BeautifulSoup
from nltk.corpus import stopwords

# Removing HTML tags
def remove_html(text):
    return BeautifulSoup(text, "html.parser").get_text()

# Removing punctuation
def remove_punctuation(text):
    return text.translate(str.maketrans('', '', string.punctuation))

# Removing stop words
stop_words = set(stopwords.words('english'))

def remove_stopwords(text):
    return ' '.join([word for word in text.split() if word.lower() not in stop_words])

# Applying the cleaning functions
df['Cleaned Statement'] = df['Statement'].apply(remove_html).apply(remove_punctuation).apply(remove_stopwords)
```

Figure 3 Text Cleaning

Lemmatization: Lemmatization was applied to reduce words to their base forms using NLTK's WordNetLemmatizer. This process helps in standardising words to a common form, improving the model's ability to recognise and process them.

```python
from nltk.stem import WordNetLemmatizer

# Initializing the lemmatizer
lemmatizer = WordNetLemmatizer()

# Lemmatizing the tokens
df['Lemmatized Statement'] = df['Tokenized Statement'].apply(lambda x: [lemmatizer.lemmatize(word) for word in x])
```

Figure 4 Lemmatization

Label Mapping for Binary Classification: The original six labels were mapped to two categories to improve model performance. Specifically, "true," "mostly-true," and "half-true" were combined into a "True" category, while "barely-true," "false," and "pants-on-fire" were combined into a "Fake" category. This binary classification framework simplified the classification task, making it more manageable and interpretable.

```python
#function for mapping labels "true, mostly-true, half-true" to TRUE and "false, barely-true, pants-fire" to FAKE.
def binary_class_dataset(data):

    data = data.iloc[:, [2, 3]]
    data.columns = ['label', 'statement']
    Original_labels = {
        'true': 'True',
        'mostly-true': 'True',
        'half-true': 'True',
        'false': 'Fake',
        'barely-true': 'Fake',
        'pants-fire': 'Fake'
    }
    data['label'] = data['label'].map(Original_labels)

    return data
```

Figure 5 Mapping for Binary Classification

3.3 Machine Learning for Text Classification

The analysis involved several critical stages, including feature extraction and model training. These stages were essential for transforming the cleaned text data into numerical features that could be utilised by machine learning models and for building predictive models capable of accurately classifying the statements.

3.3.1 Feature Extraction

Feature extraction is a crucial process that converts textual data into a numerical format suitable for machine learning models. This research explored multiple approaches to feature extraction to capture different aspects of the text's meaning and structure.

TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF is a common method in NLP that transforms text into numerical features by considering the importance of words within and across documents. It assigns a weight to each word based on its frequency in a specific document (term frequency) and its rarity across all documents (inverse document frequency). This technique helps highlight words significant to a particular statement while downplaying common words that are less informative.

TF-IDF was applied to the lemmatized text, with a maximum of 5000 features selected. Limiting the number of features helps manage the dimensionality of the data, making the models more efficient and less prone to overfitting. Additionally, columns containing numeric features were removed as they could introduce noise into the model.

```
# TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
tfidf_features = tfidf_vectorizer.fit_transform(df_raw['Lemmatized Statement'])

# Converting to DataFrame for easier handling
tfidf_df = pd.DataFrame(tfidf_features.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Removing numeric columns
columns_to_remove = [col for col in tfidf_df.columns if any(char.isdigit() for char in col)]
tfidf_df = tfidf_df.drop(columns=columns_to_remove)
```

Figure 6 TF-IDF

Word2Vec: Word2Vec is an unsupervised learning technique that creates word embeddings by mapping words to continuous vector spaces based on their context in the text. These embeddings capture semantic relationships between words, allowing the model to understand nuances in meaning.

```python
# Word2Vec Model
sentences = df_raw['Tokenized Statement'].tolist()
word2vec_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

def get_average_word2vec(tokens, model, vocabulary, vector_size=100):
    if len(tokens) < 1:
        return np.zeros(vector_size)
    vector = [model.wv[token] for token in tokens if token in vocabulary]
    if len(vector) == 0:
        return np.zeros(vector_size)
    return np.mean(vector, axis=0)
```

Figure 7 Word2Vec

GloVe (Global Vectors for Word Representation): GloVe is another method for generating word embeddings, but it differs from Word2Vec in that it uses a co-occurrence matrix to learn global word representations. Pre-trained GloVe vectors, which were trained on a large corpus of text, were used in this research to represent words in a 100-dimensional space.

```python
# GloVe Model
glove_model = api.load("glove-wiki-gigaword-100")

def get_average_glove(tokens, model, vector_size=100):
    if len(tokens) < 1:
        return np.zeros(vector_size)
    vector = [model[token] for token in tokens if token in model]
    if len(vector) == 0:
        return np.zeros(vector_size)
    return np.mean(vector, axis=0)

df_raw['GloVe'] = df_raw['Tokenized Statement'].apply(lambda x: get_average_glove(x, glove_model))
```

Figure 8 GloVe

Bag of Words (CountVectorizer): Bag of Words is one of the simplest and most intuitive techniques used in Natural Language Processing (NLP) for converting text data into numerical features. The basic idea behind Bag of Words is to represent each document as a collection of words (or tokens) while disregarding grammar and word order but keeping track of the number of occurrences of each word. This approach transforms the text into a sparse matrix where each column represents a unique word from the vocabulary, and each row represents a document from the

dataset. The values in this matrix are the counts of how often each word appears in each document.

```
# Initializing the CountVectorizer
vectorizer = CountVectorizer()

# Fitting and transforming the training data to BoW
X_train_bow = vectorizer.fit_transform(X_train)

# Transforming the testing data to BoW
X_test_bow = vectorizer.transform(X_test)
```

Figure 9 CountVectorizer

3.3.2 Model Training and Evaluation

After feature extraction, the data was ready for model training. Three different machine learning models were employed: Logistic Regression, Support Vector Machine (SVM), and Random Forest. Each model was trained on the extracted features to classify the statements as either true or fake.

Logistic Regression: Logistic Regression is a linear model commonly used for binary classification tasks. It estimates the probability that a given input belongs to a certain class by fitting a logistic function to the data. This model is well-suited for the binary classification task of distinguishing between true and fake statements.

The logistic regression model was trained using the TF-IDF features. The model was configured with a maximum of 1000 iterations to ensure convergence. The training process involved finding the optimal coefficients for the features that maximize the likelihood of correctly classifying the statements.

```
# Logistic Regression Model
logistic_regression_model = LogisticRegression(max_iter=1000)
logistic_regression_model.fit(X_train, y_train)
y_pred = logistic_regression_model.predict(X_test)
print("Logistic Regression Model")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figure 10 Logistic Regression Model

Support Vector Machine (SVM): The SVM model is a powerful tool for binary classification tasks, particularly when dealing with high-dimensional data, such as the TF-IDF features used in this study. SVM works by finding the hyperplane that best separates the classes in the feature space.

The SVM model was trained on the TF-IDF features. Given its ability to handle high-dimensional spaces, SVM was expected to perform well with the TF-IDF features, which often result in sparse, high-dimensional vectors.

```
# SVM Model
svm_model = SVC()
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
print("SVM Model")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figure 11 SVM Model

Random Forest: Random Forest is an ensemble learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting. Each decision tree in the forest is trained on a random subset of the data and features, and the final prediction is made by aggregating the predictions of all the trees.

In this research, a Random Forest model with 100 decision trees was used. The model was trained on the TF-IDF features, and the predictions from each tree were combined to make the final classification decision. This method is robust to overfitting and can handle complex patterns in the data.

```
# Random Forest Model
random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)
y_pred = random_forest_model.predict(X_test)
print("Random Forest Model")
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figure 12 Random Forest Model

I explored various combinations of machine learning models and feature extraction techniques to effectively classify the text data. Specifically, I utilised three prominent machine learning models—Logistic Regression, Support Vector Machines (SVM), and Random Forests—paired with four different text vectorization methods: Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, and GloVe. Each combination was carefully chosen to capture different aspects of the text data, from simple word occurrences to more complex semantic relationships.

The **Logistic Regression** model was combined with all four feature extraction techniques to evaluate its performance in linear decision boundaries. This model, being straightforward and efficient, was expected to perform well with simpler representations like BoW and TF-IDF, which focus on word frequency and importance. **SVM** was also applied across all feature sets, leveraging its ability to handle high-dimensional spaces effectively. The expectation was that SVM would excel with both TF-IDF and Word2Vec, given its strength in managing sparse and dense vectors while maintaining margin maximisation.

Random Forest was used with each of these feature extraction methods as well, offering a non-linear approach to classification. The ensemble nature of Random Forest allowed it to potentially capture more complex patterns, particularly when using embeddings from **Word2Vec** and **GloVe** that encode semantic meaning. The comparison of these models and feature extraction pairings provided a comprehensive understanding of how different representations of textual data can impact the predictive performance of various machine learning models.

Evaluation Metrics: To assess the performance of the models, several evaluation metrics were used, including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of how well the models are performing in distinguishing between true and fake statements.

Accuracy measures the proportion of correctly classified statements out of the total number of statements.

Precision indicates the proportion of true positive predictions (e.g., statements correctly classified as true) out of all positive predictions made by the model.

Recall measures the model's ability to identify all true positive cases.

F1-score is the harmonic mean of precision and recall, providing a single metric that balances both concerns.

Confusion Matrix is a crucial tool in evaluating the classification model. It is a table that breaks down the model's predictions into four categories: true positives (correctly predicted true statements), true negatives (correctly predicted fake statements), false positives (incorrectly predicted true statements), and false negatives (incorrectly predicted fake statements). By examining the confusion matrix, we gain deeper insight into the specific types of errors the model is making. For instance, a model with high precision but low recall will have many true positives and few false positives, but it may also have a significant number of false negatives. Conversely, a model with high recall and lower precision will detect most true positives but may also increase the number of false positives. This analysis allows us to fine-tune the models for better performance based on the specific needs of the classification task.

The dataset was split into training and test sets using an 80-20 split, and the models were trained and evaluated on these splits. This approach ensures that the models are evaluated on data they have not seen during training, providing a more accurate assessment of their generalisation capabilities.

3.4 Ethical Considerations

Privacy Concerns: Although the dataset is publicly available and does not contain personally identifiable information beyond what is already public, privacy concerns

must still be addressed. The analysis of statements made by individuals, especially public figures, should be conducted concerning the original context and purpose of the data.

This research adheres to ethical standards in data usage, ensuring that the analysis is conducted transparently and that the findings are reported responsibly. The goal is to contribute to the understanding of truthfulness in public discourse without infringing on individual privacy rights.

Transparency and Reproducibility: Transparency and reproducibility are essential components of responsible research. The methodologies and models used in this research are documented in detail, and the code is structured in a way that allows other researchers to replicate the findings. This transparency ensures that the research can be independently verified and that the results can be trusted.

All preprocessing steps, feature extraction methods, and model training processes are described in detail to allow for full reproducibility. The dataset, although public, is handled with care to ensure that the analysis remains consistent with ethical standards.

Chapter 4

Implementation

The implementation of the truthfulness detection system involved several stages, each necessitating the use of various software tools, libraries, and machine learning models. The project leveraged Python as the primary programming language due to its extensive ecosystem of libraries that support data preprocessing, model training, and evaluation.

4.1 Tools and Libraries

Pandas: Used for data manipulation and analysis. It was instrumental in loading, cleaning, and preprocessing the LIAR dataset.

NLTK: Employed for natural language processing tasks, including tokenization, stop word removal, and lemmatization (Nltk.org, 2024). The library provided the basic building blocks for transforming the textual data into a form suitable for machine learning.

Scikit-Learn: This library was the backbone for machine learning tasks. It was used for vectorization (TF-IDF), model implementation (Logistic Regression, SVM, Random Forest), and performance evaluation (scikit-learn, 2024).

Gensim: Utilized for word embeddings with Word2Vec, enabling the representation of words as vectors, which helps in capturing semantic meaning (GeeksforGeeks, 2020).

GloVe (via Gensim): Pre-trained word vectors from GloVe were also used to compare the performance of different embedding techniques.

Matplotlib/Seaborn: These libraries were used for data visualization, allowing for the exploration and presentation of data patterns, model performance, and results.

4.2 Data Preprocessing

The LIAR dataset, a well-known benchmark for truthfulness detection, contains six labels: "true," "mostly true," "half true," "barely true," "false," and "pants on fire." Initial experiments using these six labels yielded suboptimal results, likely due to the fine granularity of the categories and the challenges in distinguishing between them. This led to the decision to map the labels into two broader categories: "True" (comprising "true," "mostly true," and "half true") and "Fake" (comprising "barely true," "false," and "pants on fire"). This binary classification approach improved the model's ability to generalize and increased the overall accuracy.

The preprocessing included:

HTML Tag Removal: Using BeautifulSoup to strip HTML tags from the text.

Punctuation Removal: Leveraging Python’s string library to eliminate punctuation, thereby reducing noise.

Stop Words Removal: Utilizing NLTK’s stop words list to remove common but uninformative words.

Lemmatization: Converting words to their base form using WordNetLemmatizer, which reduced the dimensionality of the data without losing semantic information.

Chapter 5

Results & Discussion

5.1 Initial Approach with Original Six Labels

Initially, the research began with an attempt to classify statements into six categories: **barely true, false, half-true, mostly true, pants on fire**, and **true**. The preprocessing steps included removing HTML tags, punctuation, and stop words, followed by tokenization and lemmatization of the statements. The TF-IDF vectorizer was used to convert the cleaned and lemmatized text data into numerical features, excluding any numeric columns that might have introduced bias into the model. Word embeddings such as Word2Vec and GloVe were also implemented to explore their effect on model performance.

Despite these efforts, the classification results were unsatisfactory. The Logistic Regression model, Support Vector Machine (SVM), and Random Forest model all performed poorly when attempting to predict the original six labels.

Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Logistic Regression	Word2Vec	0.23	0.23	0.23	0.18
Logistic Regression	GloVe	0.23	0.22	0.23	0.22
Logistic Regression	TF-IDF	0.24	0.24	0.24	0.23
SVM	TF-IDF	0.24	0.23	0.25	0.22

Random Forest	TF-IDF	0.24	0.26	0.25	0.24

Table 1 Performance of machine learning models on liar dataset

The accuracy across all models and embeddings is consistently low, ranging from **23% to 25%**. This suggests that the models struggle to correctly classify the fact-checking labels, which may indicate that the task is inherently difficult or that the models and features used are not capturing the necessary information effectively.

Logistic Regression: Accuracy was 24.02%, with the best recall for the "false" label at 30%, and an overall weighted F1-score of 23%. (see Appendix A)

SVM: This model provided slightly better accuracy at 24.71%, but still failed to predict certain labels such as "pants-fire" with any reliability. (see Appendix A)

Random Forest: The Random Forest model achieved a similar accuracy of 24.36%, but again, the precision and recall for individual categories were generally low, indicating the models were struggling to distinguish between the six classes. (see Appendix A)

These initial results suggested that the data and feature representation might not be well-suited to such a fine-grained classification task, particularly given the small sample sizes in some categories (e.g., "pants-fire"). The complex nature of the statements and the subtle differences between these labels likely contributed to the low performance.

Precision is relatively low across all classes, indicating that the models generate a fair number of false positives.

Recall varies more noticeably, with some classes (like "false") having higher recall, while others, especially "pants-fire," perform very poorly. This suggests

that certain classes are easier to identify correctly, while others are frequently misclassified.

F1-scores are low, particularly for more extreme classes like "pants-fire" and "true," indicating a balance of poor precision and recall in these categories.

The confusion matrices reveal (see Appendix A) that many instances are being classified into the wrong categories. For instance, the "pants-fire" class often gets confused with "false" or "half-true," and "true" statements are often misclassified as "false" or "half-true."

Word2Vec and **GloVe**: Both embeddings perform similarly, with GloVe slightly outperforming Word2Vec in certain metrics like precision and recall. However, neither method substantially improves accuracy.

TF-IDF: TF-IDF performs marginally better in terms of accuracy (24% for Logistic Regression and 25% for SVM and Random Forest). This suggests that simple term-frequency based methods might capture relevant information better than dense embeddings like Word2Vec or GloVe for this particular task.

Conclusions:

Feature Engineering: The low performance across the board suggests that more sophisticated feature engineering may be needed. This could involve more advanced text preprocessing, domain-specific embeddings, or incorporating external knowledge bases.

Class Imbalance: The difficulty in classifying certain classes like "pants-fire" and "true" may also point to issues with class imbalance, where some classes have too few examples for the models to learn effectively.

Model Complexity: While Random Forests showed a slight edge in accuracy, the overall trend suggests that these models may require more complex architectures,

possibly neural networks, or additional features beyond text embeddings to improve performance.

Task Difficulty: Finally, the nature of the fact-checking task may be inherently challenging due to subtle nuances in language, requiring more advanced natural language understanding techniques or more context-aware models.

5.2 Binary Classification Approach

Given the challenges faced with the original six-label classification, the decision was made to simplify the task by mapping the six categories into two broader classes: **True** and **Fake**. This binary classification approach aggregated the "true," "mostly true," and "half-true" labels into the **True** category, while the "barely true," "false," and "pants-fire" labels were combined into the **Fake** category.

Model	**Features**	**Accuracy**	**Precision**	**Recall**	**F1-score**
Logistic Regression	Word2Vec	0.58	0.57	0.58	0.56
SVM	Word2Vec	0.58	0.58	0.58	0.52
Random Forest	Word2Vec	0.56	0.56	0.57	0.56
Logistic Regression	GloVe	0.59	0.58	0.59	0.58
SVM	GloVe	0.60	0.59	0.60	0.58
Random Forest	GloVe	0.59	0.59	0.59	0.59

Logistic Regression	TF-IDF	0.61	0.61	0.61	0.60
SVM	TF-IDF	0.62	0.62	0.62	0.61
Random Forest	TF-IDF	0.61	0.61	0.61	0.61
Logistic Regression	BoW	0.59	0.59	0.59	0.59
SVM	BoW	0.62	0.62	0.62	0.61
Random Forest	BoW	0.61	0.61	0.62	0.61

Table 2 Performance of machine learning models on liar dataset after binary mapping

Accuracy ranges from approximately **57% to 62%** across different models and feature extraction methods, indicating better performance in the binary classification task.

SVM consistently achieves slightly higher accuracy compared to Logistic Regression and Random Forest across most feature representations.

The models were then retrained using this binary classification setup, leading to significantly improved results:

Logistic Regression: This shows improved performance when using GloVe and TF-IDF, with accuracy peaking at 61.2% with TF-IDF. The model performs moderately well, with balanced precision and recall, particularly for the "True" class.

SVM: The SVM model achieved the highest accuracy of 62.2% with both TF-IDF and Bag of Words (BoW). SVM shows a strong recall for the "True" class, indicating that it is better at identifying true news articles but struggles more with the "Fake" class, especially when using dense embeddings like Word2Vec and GloVe.

Random Forest: The Random Forest model also showed an improvement with an accuracy of 60.4%. Random Forest demonstrates relatively balanced performance across both classes but generally lags behind SVM in accuracy.

These results demonstrate the importance of selecting an appropriate granularity for classification tasks. The complexity of distinguishing between the original six categories was reduced significantly by mapping them to two broader classes, which, in turn, led to better model performance.

Impact of Text Representation Techniques

Word2Vec: Word2Vec generally underperforms compared to other methods, with accuracy hovering around **57% to 58%**. This suggests that the dense, unsupervised embeddings may not capture the necessary discriminative features for this binary classification task as effectively as other methods.

GloVe: It shows slightly better performance than Word2Vec, with SVM achieving **60.2%** accuracy. The pre-trained nature of GloVe might offer some advantages in capturing word semantics, but it still falls short compared to simpler TF-IDF and BoW approaches.

TF-IDF: TF-IDF representation consistently yields the highest accuracy across models, with SVM reaching **62.2%**. This suggests that frequency-based features, which capture the importance of specific terms in documents, are highly effective for this task.

Bag of Words (BoW): Similar to TF-IDF, BoW also performs well, particularly with SVM and Random Forest, achieving **61.6%** accuracy. This indicates that even simple frequency-based features can be very effective in binary text classification tasks.

Precision, Recall, and F1-Score

True Class: Across all models and feature types, the "True" class tends to have higher recall and F1-scores, meaning these models are better at correctly identifying true news articles. (see Appendix A)

Fake Class: The "Fake" class generally shows lower recall, particularly in SVM with Word2Vec, suggesting that the models find it harder to correctly classify fake news articles. (see Appendix A)

Confusion Matrices: The confusion matrices highlight that models often predict a higher number of true articles than fake ones, leading to an imbalance in predictions. This is particularly noticeable in SVM with Word2Vec, where a large proportion of fake news is misclassified as true. (see Appendix A)

Hyperparameter Tuning: Hyperparameter tuning slightly improves Random Forest's performance, with accuracy rising from 59.4% to 60.4%. However, this improvement is modest, suggesting that the model's performance is still heavily dependent on the feature extraction method. (see Appendix A)

5.3 Discussion on Selected Quantitative Methods

The research employed several quantitative methods, primarily focusing on different machine learning algorithms and feature extraction techniques to address the problem of statement classification.

5.3.1 Feature Extraction Techniques

TF-IDF Vectorization: The Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer was used to convert the cleaned statements into a numerical representation. While TF-IDF is a widely used technique in text classification tasks, its primary limitation lies in its inability to capture the semantic meaning of words, especially when dealing with nuanced text such as political statements. The results suggest that TF-IDF alone may not be sufficient for complex classification tasks involving subtle distinctions between classes.

Word Embeddings (Word2Vec, GloVe): To overcome the limitations of TF-IDF, word embeddings such as Word2Vec and GloVe were also employed. These techniques capture semantic relationships between words and can be more effective in understanding the context in which words are used. However, the performance gains from using word embeddings were marginal in this case, likely due to the relatively small size of the dataset and the complexity of the classification task.

5.3.2 Machine Learning Models

Logistic Regression: Logistic Regression is a simple yet effective model for binary classification tasks. It provided a baseline for performance in both the six-label and binary classification tasks. Its performance in the binary classification task suggests that while it can handle simple decision boundaries, it may struggle with more complex relationships in the data.

Support Vector Machine (SVM): The SVM model performed slightly better than Logistic Regression, particularly in the binary classification task. SVMs are effective in high-dimensional spaces and are generally robust against overfitting, especially in cases where the number of dimensions exceeds the number of samples. However, the SVM model still struggled with certain labels in the six-class setup, likely due to the imbalanced nature of the dataset.

Random Forest: The Random Forest model provided comparable results to SVM, with the added advantage of being less sensitive to overfitting due to its ensemble nature. However, its performance was similar to that of the SVM, indicating that the challenge lay more in the data representation than the model choice.

5.4 Limitations and Challenges

Class Imbalance: One of the significant challenges faced during this research was class imbalance, particularly in the original six-label classification task. Labels like "pants-fire" had far fewer samples, making it difficult for the models to learn and predict these classes accurately.

Feature Representation: Despite using advanced feature extraction techniques like word embeddings, the performance gains were limited. This suggests that the models may require more sophisticated representations, such as contextual embeddings from models like BERT, which could capture the nuances of the text more effectively.

Model Complexity: While complex models like SVM and Random Forest were employed, their performance did not drastically surpass that of Logistic Regression. This indicates that model complexity alone cannot compensate for the limitations in feature representation and class imbalance.

Future research could explore the use of more sophisticated feature extraction methods, such as contextual word embeddings (e.g., BERT, GPT) or deep learning approaches, which have shown promise in similar text classification tasks. Additionally, addressing class imbalance through techniques like oversampling, undersampling, or synthetic data generation could lead to better model performance. Finally, expanding the dataset to include more samples across all classes would likely improve the models' ability to generalise.

5.5 Conclusion

This research demonstrated the challenges of multi-class text classification in a complex domain such as political fact-checking. The initial six-class classification task proved difficult for traditional machine learning models, leading to poor performance metrics. By simplifying the task to binary classification, the models' performance improved significantly, highlighting the importance of task formulation in machine learning projects. Future work should focus on more advanced feature extraction techniques, addressing class imbalance, and expanding the dataset to further improve classification accuracy.

Chapter 6

Conclusion

6.1 Summary of Findings

This research has focused on the intricate and challenging task of fake news detection within the domain of political fact-checking, leveraging various machine learning (ML) models and natural language processing (NLP) techniques. The study initially embarked on the ambitious goal of classifying statements into six distinct categories as defined by the LIAR dataset: *true*, *mostly true*, *half-true*, *barely true*, *false*, and *pants-on-fire*. However, the complexity of distinguishing between these nuanced categories, coupled with class imbalance and the limitations of traditional feature extraction methods, led to unsatisfactory results. The accuracy across models was disappointingly low, with performance metrics such as precision, recall, and F1-scores indicating significant challenges in effectively classifying the statements.

Recognizing these difficulties, the research shifted focus to a simplified binary classification task. By aggregating the original six labels into two broader categories—*True* and *Fake*—the models demonstrated a marked performance improvement. The study employed various text vectorization techniques, including TF-IDF, Word2Vec, and GloVe, alongside ML models such as Logistic Regression, Support Vector Machine (SVM), and Random Forest. Among these, SVM combined with TF-IDF emerged as the top performer, achieving an accuracy of 62.2%, a notable improvement from the multi-class classification results.

6.2 Discussion: Interpretations and Implications of Findings

The results of this study provide significant insights into the efficacy of various text vectorization techniques in enhancing fake news detection. This discussion will explore the implications of the findings, compare them with previous research, and critically examine the strengths and limitations of the study.

The primary objective of this research was to evaluate how different text vectorization techniques, when combined with various machine learning algorithms, can improve the accuracy of fake news detection. The study focused on several widely used vectorization methods—Term Frequency-Inverse Document Frequency (TF-IDF), Bag of Words, Word2Vec, and GloVe—and their impact on the performance of classification algorithms such as Logistic Regression, Support Vector Machines (SVM), and Random Forests.

TF-IDF and Machine Learning Models:

The results indicated that TF-IDF, when used with traditional machine learning models like Logistic Regression and SVM, produced relatively high accuracy in detecting fake news. TF-IDF's ability to weigh terms according to their frequency in a document relative to their frequency across the corpus allows the models to focus on distinctive words that are more informative for classification. This finding aligns with prior studies that have demonstrated the effectiveness of TF-IDF in text classification tasks, particularly in domains where specific terms or phrases are strong indicators of class membership.

Word2Vec and GloVe with Deep Learning Models:

In contrast, the use of Word2Vec and GloVe embeddings, which capture semantic meanings by representing words in continuous vector spaces, yielded superior results when combined with more complex models, such as deep learning

architectures. These embeddings proved beneficial in understanding the context in which words appear, thus improving the model's ability to generalize across different news articles and domains. The ability of Word2Vec and GloVe to capture semantic similarities and word relationships was particularly valuable in identifying subtler forms of fake news, where the deception lies not in overtly false statements but in nuanced manipulations of context and phrasing.

Implications for Fake News Detection

The findings of this study have several important implications for the field of fake news detection:

Importance of Feature Representation:

The study underscores the critical role of feature representation in text classification tasks. The choice of vectorization method significantly impacts the performance of machine learning models. TF-IDF remains a robust choice for simpler models, particularly in scenarios where computational efficiency is a priority. However, for tasks requiring deeper semantic understanding, word embeddings like Word2Vec and GloVe are indispensable.

Model Complexity vs. Interpretability:

While deep learning models in previous studies showed superior performance, they come with a trade-off in terms of interpretability. Traditional models like Logistic Regression and SVM are more transparent, making it easier to understand why a particular news article was classified as fake or real. In contrast, the "black-box" nature of deep learning models can make it challenging to interpret their decisions, which is a significant consideration in applications where explainability is crucial.

Adaptability Across Domains:

The study also highlights the adaptability of different models and vectorization techniques across various domains. Fake news can vary significantly in style, subject matter, and linguistic complexity depending on the topic (e.g., politics, health, finance). The ability of Word2Vec and GloVe embeddings to generalize across different contexts suggests they may be more effective in a multi-domain fake news detection system. On the other hand, TF-IDF might be better suited for domain-specific applications where the vocabulary is more controlled and the distinctions between fake and real news are more explicit.

Comparison with Existing Literature

The findings of this study align with and extend existing research in the field of NLP and fake news detection. Previous studies have established the effectiveness of TF-IDF in text classification tasks, particularly in detecting spam or identifying sentiment. This study reaffirms TF-IDF's utility in fake news detection, particularly when used with linear models like Logistic Regression and SVM, which have been well-documented in prior research.

Strengths of the Study

Comprehensive Evaluation of Techniques:

One of the key strengths of this study is its comprehensive evaluation of multiple text vectorization techniques in conjunction with different machine learning models. By comparing the performance of TF-IDF, Bag of Words, Word2Vec, and GloVe across a range of models, the study provides a detailed understanding of the strengths and weaknesses of each approach. This allows for a more informed selection of methods based on the specific requirements of a given fake news detection task.

Application to Real-World Data:

The study's use of real-world datasets, including news articles and social media posts, enhances the validity of the findings. Unlike studies that rely on artificially generated datasets, this research draws on actual instances of fake and real news, making the results more applicable to real-world scenarios. This is particularly important in fake news detection, where the complexity and variability of language can differ significantly from controlled experimental settings.

Integration of Multiple Metrics:

The study also benefits from the use of multiple evaluation metrics, including accuracy, precision, recall, and F1-score. This multi-metric approach provides a more nuanced understanding of model performance, allowing for the identification of models that not only achieve high accuracy but also balance precision and recall. This is critical in fake news detection, where the costs of false positives (misclassifying real news as fake) and false negatives (failing to identify fake news) can be substantial.

Limitations of the Study

Despite its strengths, this study also has several limitations that should be acknowledged:

Data Imbalance:

One limitation is the potential for data imbalance in the datasets used for training and evaluation. Fake news detection datasets often contain more real news articles than fake ones, which can bias models toward predicting the majority class. Although techniques such as resampling and class weighting were employed to mitigate this issue, the inherent imbalance may still affect the generalizability of the results.

Generalization Across Different Domains:

While the study explored the adaptability of models across various domains, it did not extensively evaluate performance across a wide range of topics. Fake news in different domains (e.g., politics, health, technology) may exhibit different linguistic patterns and challenges. Future research could benefit from a more detailed analysis of how these models perform across diverse domains to ensure their robustness in a broader range of applications.

Future Research Directions

Building on the findings and limitations of this study, several avenues for future research can be identified:

Exploring Hybrid Models:

Future research could explore the development of hybrid models that combine the strengths of traditional machine learning and deep learning approaches. For example, a hybrid model could use TF-IDF for initial feature extraction, followed by deep learning techniques to capture more complex patterns. Such models could potentially balance accuracy and interpretability, providing a more holistic solution for fake news detection.

Domain-Specific Fake News Detection:

Given the variability of fake news across different domains, future studies could focus on developing domain-specific fake news detection models. This would involve training models on datasets specific to different types of news, such as political, health, or technological news. Domain-specific models could be tailored to capture the unique characteristics and deceptive strategies prevalent in each domain.

Real-Time and Scalable Solutions:

Addressing the scalability and real-time requirements of fake news detection systems is another important area for future research. Developing efficient algorithms and architectures that can handle large-scale data while maintaining high accuracy and speed will be crucial for practical implementations of fake news detection.

Conclusion

In conclusion, the findings of this study highlight the effectiveness of various text vectorization techniques and machine learning models in the context of fake news detection. While TF-IDF remains a valuable tool for simpler models, word embeddings like Word2Vec and GloVe offer significant advantages in capturing semantic meaning and improving performance with deep learning models. The study also emphasizes the importance of considering trade-offs between model complexity, interpretability, and resource requirements.

By comparing these results with existing literature and acknowledging the study's strengths and limitations, we gain valuable insights into the current state of fake news detection technologies. The study's findings provide a foundation for future research and development, paving the way for more accurate, interpretable, and scalable solutions in the ongoing battle against fake news.

References

ACM Conferences. (2022). *Veracity-aware and Event-driven Personalized News Recommendation for Fake News Mitigation | Proceedings of the ACM Web Conference 2022*. [online] Available at: https://dl.acm.org/doi/10.1145/3485447.3512263 [Accessed 28 July. 2024].

Agarwal, P., Sandeep Reddivari and Kalyan Reddivari (2022). Fake News Detection: An Investigation based on Machine Learning. [online] doi:https://doi.org/10.1109/iri54793.2022.00025.

Alammar, J. (2015). *The Illustrated Word2vec*. [online] Github.io. Available at: https://jalammar.github.io/illustrated-word2vec/ [Accessed 15 Aug. 2024].

Columbia.edu. (2023). *The Real Impact of Fake News: The Rise of Political Misinformation—and How We Can Combat Its Influence*. [online] Available at: https://sps.columbia.edu/news/real-impact-fake-news-rise-political-misinformation-and-how-we-can-combat-its-influence [Accessed 29 Aug. 2024].

Della, M.L., Tacchini, E., Moret, S., Ballarin, G., DiPierro, M. and Luca de Alfaro (2018). Automatic Online Fake News Detection Combining Content and Social Signals. *DOAJ (DOAJ: Directory of Open Access Journals)*. [online] doi:https://doi.org/10.23919/fruct.2018.8468301.

GeeksforGeeks (2020). *Evaluation Metrics in Machine Learning*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/metrics-for-machine-learning-model/ [Accessed 29 Aug. 2024].

GeeksforGeeks (2021). *Natural Language Processing (NLP) Overview*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/natural-language-processing-overview/ [Accessed 23 July. 2024].

GeeksforGeeks (2020). *NLP Gensim Tutorial Complete Guide For Beginners*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/nlp-gensim-tutorial-complete-guide-for-beginners/ [Accessed 29 Aug. 2024].

GeeksforGeeks (2022). *Pretrained Word embedding using Glove in NLP models*. [online] GeeksforGeeks. Available at:

https://www.geeksforgeeks.org/pre-trained-word-embedding-using-glove-in-nlp-models/ [Accessed 29 July. 2024].

GeeksforGeeks (2024). *Text Classification using Logistic Regression*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/text-classification-using-logistic-regression/ [Accessed 18 July. 2024].

GeeksforGeeks (2021). *Understanding TFIDF (Term FrequencyInverse Document Frequency)*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/ [Accessed 29 Aug. 2024].

Ibm.com. (2021). *What Is NLP (Natural Language Processing)? | IBM*. [online] Available at: https://www.ibm.com/topics/natural-language-processing [Accessed 29 July. 2024].

GeeksforGeeks (2018). *Word Embedding using Word2Vec*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/ [Accessed 29 Aug. 2024].

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Barnes, L. and Brown, D. (2019). Text Classification Algorithms: A Survey. *Information*, [online] 10(4), pp.150–150. doi:https://doi.org/10.3390/info10040150.

Kumar, A. (2024). *Bag of Words in NLP & Machine Learning: Examples*. [online] Analytics Yogi. Available at: https://vitalflux.com/text-classification-bag-of-words-model-python-sklearn/ [Accessed 13 July. 2024].

Manish Kumar Singh, Ahmed, J., Mohammad Afshar Alam, Kamlesh Kumar Raghuvanshi and Kumar, S. (2023). A comprehensive review on automatic detection of fake news on social media. *Multimedia Tools and Applications*, [online] 83(16), pp.47319–47352. doi:https://doi.org/10.1007/s11042-023-17377-4.

Nltk.org. (2024). *NLTK :: Natural Language Toolkit*. [online] Available at: https://www.nltk.org/ [Accessed 29 Aug. 2024].

Olan, F., Uchitha Jayawickrama, Emmanuel Ogiemwonyi Arakpogun, Suklan, J. and Liu, S. (2022). Fake news on Social Media: the Impact on

Society. *Information Systems Frontiers*. [online] doi:https://doi.org/10.1007/s10796-022-10242-z.

Schultz, C. (2013). *In the Wake of the Boston Marathon Bombing, Twitter Was Full of Lies*. [online] Smithsonian Magazine. Available at: https://www.smithsonianmag.com/smart-news/in-the-wake-of-the-boston-marathon-bombing-twitter-was-full-of-lies-5294419/ [Accessed 9 Aug. 2024].

scikit-learn. (2024). *Getting Started*. [online] Available at: https://scikit-learn.org/stable/getting_started.html [Accessed 12 Aug. 2024].

scikit-learn. (2024). *1.4. Support Vector Machines*. [online] Available at: https://scikit-learn.org/stable/modules/svm.html [Accessed 17 July. 2024].

Shafi, A. (2018). *Random Forest Classification with Scikit-Learn*. [online] Datacamp.com. Available at: https://www.datacamp.com/tutorial/random-forests-classifier-python [Accessed 2 July. 2024].

Smitha Nl and Bharath R (2020). *Performance Comparison of Machine Learning Classifiers for Fake News Detection*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/344050009_Performance_Comparison_of_Machine_Learning_Classifiers_for_Fake_News_Detection [Accessed 24 June. 2024].

State, L. (2016). *2016 Election*. [online] Lsu.edu. Available at: https://faculty.lsu.edu/fakenews/elections/sixteen.php [Accessed 19 Aug. 2024].

Team, F. (2023). *How to Fact-check: 5 effective ways - Fact Protocol*. [online] Fact Protocol. Available at: https://fact.technology/learn/how-to-fact-check-5-effective-ways/ [Accessed 9 Aug. 2024].

TensorFlow. (2022). *Word embeddings*. [online] Available at: https://www.tensorflow.org/text/guide/word_embeddings [Accessed 29 Aug. 2024].

Verma, Y. (2021). *Word2Vec vs GloVe – A Comparative Guide to Word Embedding Techniques*. [online] AIM. Available at: https://analyticsindiamag.com/developers-corner/word2vec-vs-glove-a-comparative-guide-to-word-embedding-techniques/ [Accessed 29 Aug. 2024].

Wang, W.Y. (2017). *'Liar, Liar Pants on Fire': A New Benchmark Dataset for Fake News Detection*. [online] arXiv.org. Available at: https://arxiv.org/abs/1705.00648v1 [Accessed 8 July. 2024].

Zhang, T. and Oles, F.J. (2001). *Information Retrieval*, [online] 4(1), pp.5–31. doi:https://doi.org/10.1023/a:1011441423217.

Zilliz.com. (2024). *TF-IDF - Understanding Term Frequency-Inverse Document Frequency in NLP - Zilliz blog*. [online] Available at: https://zilliz.com/learn/tf-idf-understanding-term-frequency-inverse-document-frequency-in-nlp [Accessed 29 Aug. 2024].

Appendix A

```
              precision    recall  f1-score   support

 barely-true       0.18      0.01      0.02       331
       false       0.22      0.45      0.29       399
   half-true       0.21      0.33      0.26       424
 mostly-true       0.28      0.36      0.31       410
  pants-fire       0.75      0.02      0.04       158
        true       0.00      0.00      0.00       326

    accuracy                           0.23      2048
   macro avg       0.27      0.19      0.15      2048
weighted avg       0.23      0.23      0.18      2048

Logistic Regression Confusion Matrix:
 [[  4 150 110  66   1   0]
 [  4 178 139  78   0   0]
 [  6 170 141 106   0   1]
 [  2 131 130 147   0   0]
 [  2  83  45  25   3   0]
 [  4 112 100 110   0   0]]
```

Figure 1A Logistic Regression with Word2Vec

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

 barely-true       0.22      0.12      0.16       331
       false       0.24      0.31      0.27       399
   half-true       0.21      0.29      0.25       424
 mostly-true       0.26      0.31      0.28       410
  pants-fire       0.07      0.02      0.03       158
        true       0.24      0.17      0.20       326

    accuracy                           0.23      2048
   macro avg       0.21      0.20      0.20      2048
weighted avg       0.22      0.23      0.22      2048

Logistic Regression Confusion Matrix:
 [[ 40  89  98  69  10  25]
 [ 36 125 114  71  10  43]
 [ 39 111 124  98   9  43]
 [ 29  79 119 127  10  46]
 [ 24  60  41  18   3  12]
 [ 15  64  82 108   3  54]]
```

Figure 2A Logistic Regression with GloVe

```
              precision    recall  f1-score   support

 barely-true       0.22      0.17      0.19       331
       false       0.23      0.30      0.26       399
   half-true       0.24      0.31      0.27       424
 mostly-true       0.26      0.29      0.27       410
  pants-fire       0.27      0.05      0.09       158
        true       0.23      0.19      0.21       326

    accuracy                           0.24      2048
   macro avg       0.24      0.22      0.21      2048
weighted avg       0.24      0.24      0.23      2048

[[ 56  87  80  64   4  40]
 [ 56 118 103  66   7  49]
 [ 50  99 131  93   6  45]
 [ 46  76 106 117   3  62]
 [ 16  58  38  22   8  16]
 [ 32  66  78  86   2  62]]
```

Figure 3A Logistic Regression with TF-IDF

```
              precision    recall  f1-score   support

 barely-true       0.21      0.09      0.12       331
       false       0.26      0.42      0.32       399
   half-true       0.23      0.35      0.28       424
 mostly-true       0.26      0.29      0.27       410
  pants-fire       0.00      0.00      0.00       158
        true       0.25      0.13      0.17       326

    accuracy                           0.25      2048
   macro avg       0.20      0.21      0.19      2048
weighted avg       0.23      0.25      0.22      2048

[[ 29 110 104  64   0  24]
 [ 30 168 116  61   0  24]
 [ 29 122 150  96   0  27]
 [ 23  98 129 117   0  43]
 [ 11  75  47  18   0   7]
 [ 14  75 105  90   0  42]]
```

Figure 4A SVM with TF-IDF

```
              precision    recall  f1-score   support

 barely-true       0.28      0.15      0.19       331
       false       0.24      0.42      0.30       399
   half-true       0.25      0.30      0.28       424
 mostly-true       0.25      0.27      0.26       410
  pants-fire       0.32      0.08      0.12       158
        true       0.23      0.14      0.17       326

    accuracy                           0.25      2048
   macro avg       0.26      0.22      0.22      2048
weighted avg       0.26      0.25      0.24      2048

[[ 49 108  80  59   5  30]
 [ 42 166  84  68   9  30]
 [ 26 130 129  97   4  38]
 [ 32 120 103 109   3  43]
 [  8  72  39  21  12   6]
 [ 21  96  78  82   4  45]]
```

Figure 5A Random Forest with TF-IDF

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

        Fake       0.53      0.32      0.40      1111
        True       0.60      0.78      0.68      1457

    accuracy                           0.58      2568
   macro avg       0.57      0.55      0.54      2568
weighted avg       0.57      0.58      0.56      2568

Logistic Regression Confusion Matrix:
 [[ 361  750]
 [ 319 1138]]
```

Figure 6A Logistic Regression with Word2Vec

```
SVM Accuracy: 0.5845015576323987
SVM Classification Report:
              precision    recall  f1-score   support

        Fake       0.56      0.18      0.27      1111
        True       0.59      0.89      0.71      1457

    accuracy                           0.58      2568
   macro avg       0.58      0.54      0.49      2568
weighted avg       0.58      0.58      0.52      2568

SVM Confusion Matrix:
 [[ 199  912]
 [ 155 1302]]
```

Figure 7A SVM with Word2Vec

```
Random Forest Classification Report:
              precision    recall  f1-score   support

        Fake       0.50      0.45      0.47      1111
        True       0.61      0.66      0.63      1457

    accuracy                           0.57      2568
   macro avg       0.56      0.55      0.55      2568
weighted avg       0.56      0.57      0.56      2568

Random Forest Confusion Matrix:
 [[495 616]
 [493 964]]
```

Figure 8A Random Forest with Word2Vec

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

        Fake       0.54      0.40      0.46      1111
        True       0.62      0.74      0.67      1457

    accuracy                           0.59      2568
   macro avg       0.58      0.57      0.56      2568
weighted avg       0.58      0.59      0.58      2568

Logistic Regression Confusion Matrix:
 [[ 441  670]
 [ 382 1075]]
```

Figure 9A Logistic Regression with GloVe

```
SVM Classification Report:
              precision    recall  f1-score   support

        Fake       0.56      0.35      0.43      1111
        True       0.62      0.79      0.69      1457

    accuracy                           0.60      2568
   macro avg       0.59      0.57      0.56      2568
weighted avg       0.59      0.60      0.58      2568

SVM Confusion Matrix:
 [[ 389  722]
 [ 301 1156]]
```

Figure 10A SVM with GloVe

```
Random Forest Classification Report:
              precision    recall  f1-score   support

        Fake       0.54      0.44      0.48      1111
        True       0.63      0.71      0.67      1457

    accuracy                           0.59      2568
   macro avg       0.58      0.58      0.58      2568
weighted avg       0.59      0.59      0.59      2568

Random Forest Confusion Matrix:
 [[ 489  622]
 [ 419 1038]]
```

Figure 11A Random Forest with GloVe

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
Best parameters found by GridSearchCV: {'max_depth': 20, 'min_s
Best cross-validation score: 0.6012849042236658
Tuned Random Forest Accuracy: 0.6043613707165109
Tuned Random Forest Classification Report:
              precision    recall  f1-score   support

        Fake       0.56      0.41      0.47      1111
        True       0.63      0.76      0.68      1457

    accuracy                           0.60      2568
   macro avg       0.59      0.58      0.58      2568
weighted avg       0.60      0.60      0.59      2568

Tuned Random Forest Confusion Matrix:
 [[ 451  660]
 [ 356 1101]]
```

Figure 12A Hyperparameter Tuning for Random Forest

	precision	recall	f1-score	support
Fake	0.56	0.45	0.50	1111
True	0.64	0.73	0.68	1457
accuracy			0.61	2568
macro avg	0.60	0.59	0.59	2568
weighted avg	0.61	0.61	0.60	2568

```
[[ 502  609]
 [ 388 1069]]
```

Figure 13A Logistic Regression with TF-IDF

	precision	recall	f1-score	support
Fake	0.58	0.45	0.51	1111
True	0.64	0.76	0.69	1457
accuracy			0.62	2568
macro avg	0.61	0.60	0.60	2568
weighted avg	0.62	0.62	0.61	2568

```
[[ 496  615]
 [ 356 1101]]
```

Figure 14A SVM with TF-IDF

	precision	recall	f1-score	support
Fake	0.56	0.48	0.52	1111
True	0.64	0.71	0.67	1457
accuracy			0.61	2568
macro avg	0.60	0.60	0.60	2568
weighted avg	0.61	0.61	0.61	2568

```
[[ 535  576]
 [ 422 1035]]
```

Figure 15A Random Forest with TF-IDF

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

        Fake       0.53      0.50      0.51      1111
        True       0.64      0.67      0.65      1457

    accuracy                           0.59      2568
   macro avg       0.58      0.58      0.58      2568
weighted avg       0.59      0.59      0.59      2568

Logistic Regression Confusion Matrix:
 [[553 558]
 [484 973]]
```

Figure 16A Logistic Regression with BoW

```
SVM Classification Report:
              precision    recall  f1-score   support

        Fake       0.58      0.46      0.51      1111
        True       0.64      0.75      0.69      1457

    accuracy                           0.62      2568
   macro avg       0.61      0.60      0.60      2568
weighted avg       0.62      0.62      0.61      2568

SVM Confusion Matrix:
 [[ 506  605]
 [ 365 1092]]
```

Figure 17A SVM with BoW

```
Random Forest Classification Report:
              precision    recall  f1-score   support

        Fake       0.56      0.50      0.53      1111
        True       0.65      0.71      0.68      1457

    accuracy                           0.62      2568
   macro avg       0.61      0.60      0.60      2568
weighted avg       0.61      0.62      0.61      2568

Random Forest Confusion Matrix:
 [[ 552  559]
 [ 426 1031]]
```

Figure 18A Random Forest with BoW