People's Democratic Republic of Algeria

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Yahia Fares University of Medea

Faculty of Sciences

Department of Mathematics and Computer Science



Graduation Project

Submitted for Master Degree in Computer Science

Specialty: System Engineering and Web Technology

Tracking and Detecting Fake News and Deepfakes in social media

Proposed and directed by:	Presented to the jury:	Prepared and presented by:
Dr. Ahmed Cherif Mazari	Dr. Djamel Hassani	Miss.Amina Mzaad
Dr. Azzeddine Benaissa Rachid	Dr. Mohamed Yassine Hayi	Mrs.Ikram Alili

2023 /2024

Acknowledgement

We thank God Almighty for giving us the health and will to begin and complete this message.

First of all, this work would not have been as rich and would not have been possible without the help and supervision of Mr. Ahmed Cherif Mazari and Mr. Rachid Benaissa Azzeddine. We thank them for the exceptional quality of them supervision, and for them patience, rigor, and availability during our work in preparing this thesis.

We would also like to thank all our professors at our university and the mathematics department

Computer Science for their great generosity and patience that they were able to show despite their academic and professional responsibilities.

Finally, I thank my family for their patience, help and moral support throughout the completion of this work.

thank you all.

Dedication

I send my sincere thanks to Almighty Allah, the Most High, who has guided and helped us throughout our academic journey and our entire lives.

To my dear parents... for all their sacrifices, their love, their tenderness, their support, and their prayers throughout my studies.

To my dear sisters and brother... for their constant encouragement and moral support.

To my grandmothers... To the soul of my grandfathers and my paternal grandmother... To all my family.

To my friends Manel Benoua and Bouchra Abdennebi for their help and support, and to my bestie Bouchra Nourine.

To all my dear friends who have brought happiness and made a positive impact on my life.

To all my professors who have worked hard to teach me.

To myself for my diligence and dedication throughout my academic journey. I wish for more success and excellence.

AMINA MZAAD

Dedication

Above all, I thank the great God, who helped us to develop this modest work.

I have the immense honor of dedicating this modest work:

To my dear father All the ink in the world could not be enough to express my feelings towards a very dear one. You have always been my school of patience, trust and above all hope and love. You are and you will remain my reference for me, the light that illuminates my path.

To my dear mother who worked for my success, with her love, her support, all the sacrifices made and her precious advice, for all her assistance and her presence in my life, receive through this work, however modest it may be. , the expression of my feelings and my eternal gratitude.

To my husband who always believed in me, even when I doubted my abilities.

To my brother Mohammed, as well as my sisters Wissam and Hadil

To the entire Alili Family and the Khelidj Family without exception

ALILI IKRAM

Abstract

The spread of fake news and deepfakes on social media platforms has become a significant issue affecting public opinion and decision-making. Recent advances in artificial intelligence, particularly deep learning architectures, have shown promising results in addressing this problem. This report examines the effectiveness of machine learning, deep learning, and pre-trained models, including BERT, RoBERTa, FakeBERT, LSTM, Bi-LSTM, GRU, Bi-GRU, SVM, LR, and RF, for detecting fake news, and the Gemini API for deepfake detection on ISOT, AFND, and AFND TweepFake datasets. The proposed approach leverages the strengths of these models to enhance accuracy in detecting fake news and deepfakes. Evaluation metrics such as precision, recall, and F1 score are used to assess performance, where RoBERTa and FakeBERT achieved the best results for ISOT with an accuracy of 99.98%, and BERT achieved the best result for AFND with an accuracy of 92.17%.

Keywords: Fake news detection, deepfakes, deep learning, disinformation, text classification, machine learning, pre-trained.

ملخص

أصبح انتشار الأخبار الكاذبة والتزييف العميق على منصات التواصل الاجتماعي قضية هامة تؤثر على الرأي العام وصنع القرار. وقد أظهرت التطورات الحديثة في الذكاء الاصطناعي، وخاصة بنيات التعلم العميق، نتائج واعدة في حل هذه المشكلة. يتناول هذا التقرير فعالية التعلم الآلي والتعلم العميق والنماذج المدربة مسبقًا، بما في ذلك BERT وRoBERTa وFakeBERT وLSTM وBi-LSTM وGRU وBi-GRU وSVM وLR وRF للكشف عن الأخبار المزيفة، واجهة برمجة تطبيقات Gemini للكشف عن التزييف العميق في مجموعات بيانات ISOT وAFND وAFND TweepFake. يستغل النهج المقترح نقاط القوة في هذه النماذج لتحسين دقة الأخبار المزيفة والكشف عن التزييف العميق. يتم استخدام مقاييس التقييم مثل الدقة والاستدعاء ودرجة F1 لتقييم الأداء، حيث حقق RoBERTa وFakeBERT أفضل النتائج لـ ISOT بدقة 99.98%، وحقق BERT أفضل نتيجة لـ AFND بدقة 92.17%.

الكلمات المفتاحية: اكتشاف الأخبار المزيفة، التزييف العميق، التعلم العميق، المعلومات المضللة، تصنيف النص، التعلم الآلي، نقل التعلم.

Résumé

La diffusion de fausses nouvelles et de deepfakes sur les plateformes de médias sociaux est devenue un problème important qui affecte l'opinion publique et la prise de décision. Les progrès récents en matière d'intelligence artificielle, en particulier les architectures d'apprentissage profond, ont donné des résultats prometteurs pour résoudre ce problème. Ce rapport examine l'efficacité de l'apprentissage automatique, de l'apprentissage profond et des modèles pré-entraînés, notamment BERT, RoBERTa, FakeBERT, LSTM, Bi-LSTM, GRU, Bi-GRU, SVM, LR et RF, pour détecter les fausses nouvelles, et l'API Gemini pour la détection des deepfakes sur les ensembles de données ISOT, AFND et AFND TweepFake. L'approche proposée exploite les atouts de ces modèles pour améliorer la précision de la détection des fausses nouvelles et des deepfakes. Des mesures d'évaluation telles que la précision, le rappel et le score F1 sont utilisées pour évaluer les performances, où RoBERTa et FakeBERT ont obtenu les meilleurs résultats pour ISOT avec une précision de 99,98 %, et BERT ont obtenu le meilleur résultat pour l'AFND avec une précision de 92,17 %.

Mots-clés : détection de fausses nouvelles, deepfakes, deep learning, désinformation, classification de textes, apprentissage automatique, apprentissage par transfert.

Table of content

Figure List ---- i

Tables List ---- iii

Abbreviations List ---- iv

General Introduction ---- 1

Chapter 1
Fake News Detection

1.1. Introduction ---- 4

1.2. Social Media ---- 4

1.3. Fake news ---- 4

1.4. Related concept of Fake news ---- 5

1.4.1. Fabricated content ---- 6

1.4.2. Propaganda ---- 6

1.4.3. Hoax ---- 6

1.4.4. Rumor ---- 6

1.4.5. Satire news/ Parody News ---- 6

1.4.6. Junk News ---- 7

1.4.7. Manipulated Visual Content ---- 7

1.4.8. Click-bait ---- 7

1.5. DeepFakes ---- 7

1.5.1. Visual-Based ---- 7

1.5.2. Audio-Based ---- 8

1.5.3. Text-Based ---- 8

1.6. Fake news categories ---- 9

1.6.1. Misinformation ---- 9

1.6.2. Disinformation ---- 9

1.6.3. Mal-information ---- 10

1.7. Why fake news spread in social media? ---- 10

1.8. Impact of fake news ---- 11

1.8.1. Impact on health ---- 11

1.8.2. Political impact ---- 12

1.8.3. Impact of local rumors and fake news on society ---- 13
1.8.4. Financial impact ---- 13
1.9. Fake News Detection Techniques on social media ---- 13
1.9.1. News Content-Based ---- 14
1.9.1.1. Knowledge-Based system ---- 14
1.9.1.2. Style-Based ---- 15
1.9.1.3. Linguistic-Based ---- 15
1.9.1.4. Visual-Based ---- 15
1.9.2. Social Context-Based ---- 16
1.9.2.1. Network-Based ---- 16
1.9.2.2. Temporal-Based ---- 16
1.9.2.3. Credibility-Based ---- 16
1.10. Challenges in fake news detection ---- 17
1.10.1. Variety ---- 17
1.10.2. Volume ---- 17
1.10.3. Velocity ---- 18
1.10.4. Veracity ---- 18
1.10.5. Valid data ---- 18
1.11. Conclusion ---- 19

Chapter 2
Literature Review

2.1. Introduction ---- 21
2.2. NLP ---- 21
2.2.1. DATA pre-processing ---- 21
2.2.2. Word embedding ---- 25
2.3. Machine learning ---- 30
2.3.1. Supervised Machine Learning ---- 31
2.3.1.1. Classification ---- 31
2.3.1.2. Regression ---- 32
2.3.2. Unsupervised Machine Learning ---- 33
2.3.2.1. Clustering ---- 33
2.3.2.2. Association ---- 34
2.3.3. Semi-Supervised Machine Learning ---- 35
2.4. Deep learning ---- 36
2.4.1. Deep Learning Techniques for Fake News Detection ---- 36

2.5. Deepfake Generation (deepfake) ---- 40
2.6. Related works ---- 40
2.6.1. NLP Related Works ---- 41
2.6.2. Machine learning Related Works ---- 42
2.6.3. Deep Learning Related Works: ---- 43
2.6.4. Deepfakes Related Works ---- 45
2.6.5. Datasets ---- 45
2.7. Conclusion ---- 50

Chapter 3
Proposed Approaches

3.1. Introduction ---- 52
3.2. Methodology ---- 52
3.3. Proposed Approach ---- 53
3.3.1. FakeBERT ---- 53
3.3.1. LSTM ---- 54
3.3.3. GRU ---- 54
3.3.4. Bi-LSTM ---- 54
3.3.5. Bi-GRU ---- 54
3.3.6. BERT ---- 55
3.3.7. RoBERTa ---- 55
3.3.7. Logistic regression (LR) ---- 55
3.3.8. Random Forest ---- 55
3.3.9. Support Vector Machine (SVM) ---- 56
3.3.10. DeepFake Detection Approach ---- 56
3.4. Metrics Utilized for Evaluating Model Performance ---- 56
3.4.1. Confusion matrix ---- 57
A. Accuracy ---- 57
B. Precision ---- 58
C. F1-Score ---- 58
D. Recall ---- 58
3.5. Conclusion ---- 58

Chapter 4
Implemented Approaches

4.1. Introduction ---- 60

4.2. Data Used for Training and Test ---- 60

4.2.1. ISOT Fake News Dataset ---- 60

4.2.2. AFND ---- 61

4.2.3. TweepFake Dataset ---- 62

4.3. Environment and Development Tools ---- 63

4.3.1. Python ---- 63

4.3.2. Libraries ---- 64

A. For Data Manipulation ---- 64

B. For Text Processing ---- 64

C. For pre-trained models implementation ---- 65

D. For Deep Learning implementation ---- 65

E. For data Visualization ---- 66

4.3.3. Kaggle ---- 67

4.3.3.1. Session information ---- 68

4.4. Build approaches ---- 69

4.4.1. Data Preprocessing ---- 69

4.4.2. Splitting the dataset ---- 72

4.4.3. Word embedding ---- 72

4.4.4. Models building ---- 74

4.4.4.1. FakeBERT model Architecture ---- 74

4.4.4.1. LSTM Model Architecture ---- 76

4.4.4.3. GRU model Architecture ---- 79

4.4.4.4. BI-LSTM Model Architecture ---- 81

4.4.4.5. BI-GRU Model Architecture ---- 83

4.4.4.6. Pre-trained models ---- 85

4.4.4.7. Support vector machine ---- 88

4.4.4.8. Random forest ---- 89

4.4.4.9. Logistic regression ---- 90

4.4.4.10. DeepFake Detection Approach ---- 90

4.5. Conclusion ---- 92

Chapter 5
Results and Evaluation

5.1. Introduction ---- 94

5.2. Results and Discussion ---- 94

5.2.1. Training and Validation Results---------- 94
5.2.2. Comparison between proposed approaches ---------- 107
5.2.3. Comparison of proposed approach with existing works---------- 109
5.3. Web Application To detect Fake News---------- 110
5.4. Conclusion ---------- 113
General Conclusion ---------- 114
Bibliography ---------- 116
Web Links---------- 124

Figure List

Figure 1.1.*Spread of Fake News on social media.* 5
Figure 1.2.*DeepFakes Types.* 9
Figure 1.3.*Fake News Categories.* 10
Figure 1.4.*Automatic fact-checking process.* 15
Figure 1.5.*Fake News Detection Techniques on social media.* 17
Figure 1.6.*Challenges in fake news detection.* 19
Figure 2.1.*GloVe Architecture.* 27
Figure 2.2.*Word2vec architecture.* 28
Figure 2.3.*Doc2vec architecture.* 29
Figure 2.4.*Supervised Machine Learning.* 31
Figure 2.5.*Unsupervised Machine Learning.* 33
Figure 2.6.*Semi-Supervised Machine Learning.* 35
Figure 2.7.*Convolutional Neural Network (CNN).* 37
Figure 2.8.*Recurrent Neural Network (RNN).* 38
Figure 2.9.*A feedforward network with one hidden layer.* 39
Figure 2.10.*a pie chart showing the benchmark datasets used in studies on the identification of false information.* . . 47
Figure 3.1.*Proposed Approach Overview flowchart.* 53
Figure 3.2.*Confusion matrix.* 57
Figure 4.1.*ISOT Fake News Dataset.* 61
Figure 4.2.*Example of the contents of the JSON file named "sources.json".* 61
Figure 4.3.*AFND Arabic fake news dataset.* 62
Figure 4.4.*TweepFake Dataset.* 63
Figure 4.5.*import session information.* 68
Figure 4.6.*Session information.* 68
Figure 4.7.*clean the data.* 69
Figure 4.8.*lowercase the dataset.* 69
Figure 4.9.*Remove Characters.* 69
Figure 4.10.*Remove Stop words.* 70
Figure 4.11.*Stemming the dataset.* 70
Figure 4.12.*Tokenization the dataset.* 70
Figure 4.13.*call the Bert embedding.* 71
Figure 4.14.*Tokenization using RoBERTa tokenizer.* 71
Figure 4.15.*split the data into validation, training and testing.* 72
Figure 4.16.*encoding the data using tfidf.* 72
Figure 4.17.*Fake Bert Architecture.* 76
Figure 4.18.*LSTM Architecture.* 78
Figure 4.19.*GRU Architecture.* 80
Figure 4.20.*Bi-LSTM Architecture.* 82
Figure 4.21.*Bi-GRU Architecture.* 85
Figure 4.22.*Bert model.* 86
Figure 4.23.*RoBERTa model.* 86
Figure 4.24.*BERT Architecture.* 87
Figure 4.25.*Initialize the SVM model with kernel rbf or linear.* 88
Figure 4.26.*Define the parameter grid and Initialize GridSearchCV.* 88
Figure 4.27.*Get the best parameters and Train Model with them.* 88
Figure 4.28.*Initialize RandomForestClassifier.* 89
Figure 4.29.*Define RandomForest Parameter.* 89
Figure 4.30.*Get Best Parameters and Train Model.* 89
Figure 4.31.*Initialize the Logistic regression model.* 90

Figure 4.32.*Train the model.* *90*
Figure 4.33.*Train the model.* *91*
Figure 4.34.*the new data with prompted labels.* *91*
Figure 5.1.*Training and validation accuracy and loss graphs of FakeBert model with ISOT Dataset.* *94*
Figure 5.2.*Training and validation accuracy and loss graphs of LSTM model with ISOT Dataset.* *95*
Figure 5.3.*Training and validation accuracy and loss graphs of GRU model with ISOT Dataset.* *95*
Figure 5.4.*Training and validation accuracy and loss graphs of BILSTM model with ISOT Dataset* *96*
Figure 5.5.*Training and validation accuracy and loss graphs of BIGRU model with ISOT Dataset.* *96*
Figure 5.6.*Training and validation accuracy and loss graphs of Bert model with ISOT Dataset.* *97*
Figure 5.7.*Training and validation accuracy and loss graphs of RoBERTa model with ISOT Dataset.* *97*
Figure 5.8.*confusion matrix of Linear_SVM and RBF_SVM models with TFIDF word embedding with ISOT Dataset.* *98*
Figure 5.9.*confusion matrix of Linear_SVM and RBF_SVM models with BERT word embedding with ISOT Dataset.* *98*
Figure 5.10.*confusion matrix of Logistic Regression model with ISOT Dataset.* *99*
Figure 5.11.*confusion matrix of Random Forest model with ISOT Dataset.* *99*
Figure 5.12.*Training and validation accuracy and loss graphs of FakeBert model with AFND Dataset.* *100*
Figure 5.13.*Training and validation accuracy and loss graphs of LSTM model with AFND Dataset.* *101*
Figure 5.14.*Training and validation accuracy and loss graphs of GRU model with AFND Dataset.* *101*
Figure 5.15.*Training and validation accuracy and loss graphs of BILSTM model with AFND Dataset.* *102*
Figure 5.16.*Training and validation accuracy and loss graphs of BIGRU model with AFND Dataset.* *102*
Figure 5.17.*Training and validation accuracy and loss graphs of Bert model with AFND Dataset.* *103*
Figure 5.18.*Training and validation accuracy and loss graphs of RoBERTa model with AFND Dataset.* *103*
Figure 5.19.*confusion matrix of Linear_SVM anf RBF_SVM models with Bert word embedding with AFND Dataset.* *104*
Figure 5.20.*confusion matrix of Linear_SVM and RBF_SVM model with TFIDF word embedding with AFND Dataset* *104*
Figure 5.21.*confusion matrix of Logistic Regression with AFND Dataset.* *105*
Figure 5.22.*confusion matrix of Random Forest with AFND Dataset.* *105*
Figure 5.23.*compare between gemini API response and ground truth.* *107*
Figure 5.24.*Comparison of Medel Accuracies o ISOT and AFND Datasets.* *107*
Figure 5.25*.Detect Arabic Fake News.* *111*
Figure 5.26.*Detect English Fake News.* *112*

Tables List

Table 2.1.*The table presents the partitioning of the dataset based on training, validation and testing with advantages and disadvantages* 23
Table 2.2.*A brief overview of the published papers focused on fake news detection.* 47
Table 4.1.*FakeBert model Hyperparameters.* 75
Table 4.2.*LSTM model Hyperparameters* 77
Table 4.3.*GRU model Hyperparameters.* 79
Table 4.4.*BILSTM model Hyperparameters.* 81
Table 4.5.*BIGRU model Hyperparameters.* 83
Table 4.6.*Transfers models Hyperparameters.* 88
Table 4.7.*SVM model Hyperparameters.* 89
Table 4.8.*RandomForest model Hyperparameters* 90
Table 5.1.*The performance of suggested models on ISOT Fake News dataset.* 100
Table 5.2.*The performance of suggested models on AFND Fake News dataset.* 106
Table 5.3.*TweepFakes results with each scenario.* 106
Table 5.4. *Comparison of proposed approach with previous contributions results.* 109

Abbreviations List

API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BILSTM	Bidirectional LSTM
BIGRU	Bidirectional GRU
CNN	Convolutional Neural Network
CNTK	Microsoft Cognitive Toolkit
FNN	Feedforward Neural Network
GAN	Generative Adversarial Network
GLUE	General Language Understanding Evaluation
GPT	Generative Pre-trained Transformer
GRU	Gated Recurrent Unit
GPU	Graphics Processing Unit
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
LR	Logistic Regression
NLP	Natural language processing
NB	Naive Bayes
NSP	Next Sentence Prediction
RF	Random Forest
RNN	Recurrent Neural Network
RoBERTa	Robustly optimized BERT approach
SVM	Support Vector Machine
TPU	Tensor Processing Unit
URL	Uniform Resource Locator
XLM	Cross-lingual Language Model

General Introduction

Inaccurate and misleading journalism has persisted for centuries before the advent of the Internet, spreading through means such as word of mouth, print materials, radio, television, and eventually digital media. The emergence of the social media era in 1997 marked a turning point, enabling the rapid dissemination of fake news and deepfakes across social media platforms, posing significant challenges in identifying and mitigating their impacts. A recent study found that fake news stories are 70% more likely to be retweeted on Twitter than true stories (Vosoughi et al., 2018), exacerbating the difficulty of distinguishing fact from fiction. This phenomenon is amplified by the increasing complexity and volume of data circulating on these platforms, providing fertile ground for misinformation to spread rapidly and widely, with far-reaching consequences for individuals, communities, and even entire countries by manipulating public opinion, instigating fear and hatred, and undermining confidence in democratic institutions and processes.

As a result, combating disinformation has become increasingly challenging, necessitating innovative strategies and robust fact-checking mechanisms to safeguard the integrity of information ecosystems.

Researchers have developed algorithmic methods to automate the process of detecting fake news, which are more effective and less expensive than traditional methods such as manual fact-checking and crowdsourcing. Among these methods are those that rely on machine learning, deep learning, and pre-trained models. These methods have shown potential in recognizing patterns and categorizing news articles as authentic or deceptive. However, further research is needed to enhance the accuracy and reliability of these techniques.

This study aims to propose an approach for detecting fake news and deepfakes on social media. The report will investigate the effectiveness of various machine learning, deep learning, and pre-trained models, including Fakebert, LSTM (Long Short-Term Memory), BiLSTM (Bidirectional Long Short-Term Memory), GRU (Gated Recurrent Unit), BiGRU (Bidirectional Gated Recurrent Unit), BERT (Bidirectional Encoder Representations from Transformers), RoBERTa, SVM (Support Vector Machine), RF (Random Forest), LR (Logistic Regression) and Gemini API. To achieve the report's objectives, three datasets will be utilized: the ISOT Fake News dataset and the AFND Arabic fake news dataset and TweepFake dataset. These datasets will be preprocessed and used to train the selected models. Various metrics, including Accuracy, Precision, Recall, and F1-

score, will be employed to evaluate the performance of the models. Furthermore, the proposed approaches will be compared to assess their effectiveness.

The importance of this report lies in its potential to contribute to the development of more effective and efficient methods for detecting fake news.

The subsequent sections of this report are organized as follows: Chapter 1 introduces the topic of fake news, covering its terminology, consequences, and measures taken to address it. Chapter 2 discusses previous research on fake news identification. Chapter 3 presents the proposed methodology and information about evaluation metrics. The architectures of our models are presented in Chapter 4, and the results and evaluation are discussed in Chapter 5. Finally, the conclusion and future possibilities are discussed.

Chapter 1

Fake News Detection

1.1. Introduction

During the early days of the Internet, people believed that the Internet would enhance the transparency of information, which would improve the quality of democracies (Abramson et al., 1990) (Tewksbury, 2003).

However, with the development and widespread of social media the landscape has changed radically, it has becoming increasingly easy to disseminate fake news and deepfakes and target individuals with the intention of misleading them or changing their opinions. This shift has presented challenges in identifying fake news and mitigating its effects.

This chapter introduces the issue of detecting fake news in social media, discussing its adverse effects, past solutions, and challenges.

1.2. Social Media

The term 'Social Media'(SM) was first used in 1994 on Tokyo online media environment called Matisse. During the initial phase of internet, the first social media platforms were created and introduced. Over time, the number of social media platforms and active users has experienced substantial growth, solidifying its status as one of the internet's most crucial applications.

We can define social media as any online resource that is designed to facilitate engagement between individuals (Bishop, 2019).

1.3. Fake news

To understand the term "fake news," it is essential to grasp the meanings of "news" and "fake." Traditionally, news has been perceived as a journalistic product expected to deliver independent, reliable, accurate, and comprehensive information (Kovach & Rosenstiel, 2007) .

However, news is influenced by external factors such as the political establishment, readers, and advertisers (Shoemaker & Reese, 2013).

The term "fake" is associated with forged, counterfeit, unauthentic, imitation, deceitful, dishonest, misleading, and fraudulent practices (Abbasi et al., 2010).

Fake news, as defined by Leonhardt in The New York Times[i], pertains to the intentional dissemination of false information through both traditional and online media platforms. This false

information is often characterized as yellow and misleading propaganda or journalism. (Allcott & Gentzkow, 2017) further elaborate that fake news involves intentionally and verifiably false articles with the capacity to mislead readers. (Mavridis, 2018) extends this definition by highlighting that fake news can serve various purposes, including spreading propaganda, influencing political opinions, or being created for entertainment or profit.

(Gelfert, 2018) identifies common features in different definitions, underscoring the role of social media in the creation and dissemination of fake news, the scarcity of accurate information, and the deliberate intention to mislead readers. Gelfert proposes a specific definition, asserting that fake news entails the intentional dissemination of false or deceptive information presented as legitimate news, designed with the specific intent to mislead.

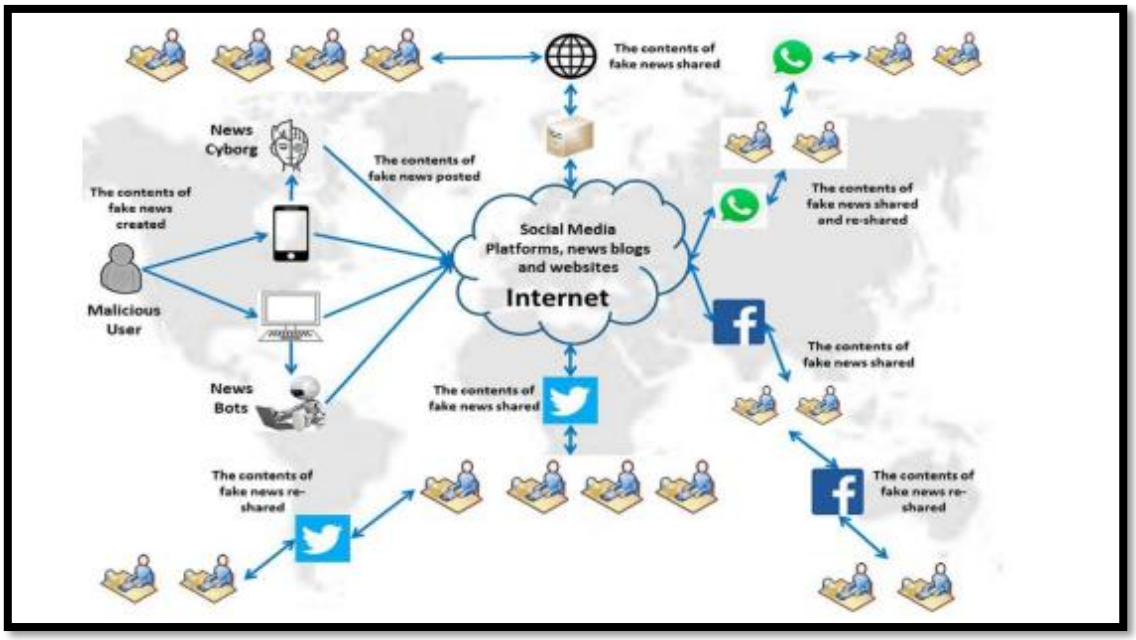


Figure 1.1.*Spread of Fake News on social media.*

1.4. Related concept of Fake news

Fake news includes various forms and methodologies, each with distinct characteristics and purposes. Understanding these categories helps in identifying and combating misinformation effectively.

1.4.1.Fabricated content

(Burkhardt, 2017) called this category of fake news "canard," as it represents a form of fake news widely spread on social media. This type includes completely fabricated content that has no basis in reality, and is designed to mislead for bad motives. Fabricated content lack any reference to fiction, which makes them dangerous by being indistinguishable from real news.

1.4.2. Propaganda

It is defined as news that has some elements of factual stripes, but are deliberately false, twisted, or exaggerated to a point where it does not reflect reality (Reilly, 2018).

In another meaning, propaganda refers to creating news stories with strong biases to an entity. This entity could be a political party, public figure, organization, government, or a cause. Propagandists may fabricate facts, be selective with facts, and present opinionated facts (Edgar & Sedgwick, 2007).

1.4.3. Hoax

It defines as a lie created with the aim of passing itself off as the truth (MacDougall, 1958) .

1.4.4. Rumor

A claim which did not originate from news events and that has not been verified while it spreads from one person to another. According to (Shu et al., 2017), Rumor is an unverified or doubtful story that spreads from person to another.

1.4.5. Satire news/ Parody News

It is factual news presented using humor, irony, exaggeration, ridicule, and mockery (Brodie, 2018).

(Brunvand, 1963) defined parody as "jokes that defraud the listener into believing that he is hearing a true narrative and then suddenly turn out to be all nonsense"

Satire news are written with the primary purpose of entertaining or criticizing the readers, but similarly to hoaxes they can be harmful when shared out of context. They are characterized by humor, irony and absurdity and they can mimic genuine news.

1.4.6. Junk News

Junk News is a broad concept that typically includes news content associated with a post, rather than focusing on a single article (Woolley & Howard, 2018).

1.4.7. Manipulated Visual Content

This category includes the manipulation of images or videos to present false evidence as factual support for a claim. It's viral on social media. Instances include fake screenshots or images presented out of context. (Safieddine & Ibrahim, 2020)

1.4.8. Click-bait

It is a type of low-quality journalism that focuses on attracting followers and generating revenue by using sensationalist headlines and content containing exaggerated or misleading information to attract clicks. This practice prioritizes attention-grabbing tactics over providing accurate and objective information. (Volkova et al., 2017)(Chen et al., 2015)

1.5. DeepFakes

Deepfake media, in its strictest sense, refers to the application of artificial intelligence and deep learning techniques to manipulate or synthesis multimedia content, specifically visual (images and videos), audio, and text.[ii]

These techniques involve producing exceptionally realistic and frequently misleading content that poses challenges in discerning its authenticity from genuine media (Ma et al., 2017).

Deepfakes can be categorized into three primary types depending on the primary medium they alter: visual, audio, and text-based deepfakes. Each category involves distinct manipulation techniques, as depicted in Figure1-2, which illustrates the various types and their respective applications.

1.5.1. Visual-Based

Visual deepfakes encompass the manipulation or synthesis of images and videos through deep learning-based techniques. These methods alter facial expressions, gestures, lip movements, and even entire body movements of individuals, enabling them to appear to say or do things they never actually did.

Moreover, these deepfakes can facilitate impersonation by superimposing the faces of individuals onto someone else's body in videos and images.

1.5.2. Audio-Based

Audio deepfakes are generated employing artificial intelligence and machine learning methodologies. This process entails training a deep learning model on a dataset consisting of audio recordings of a particular individual's voice.

This dataset aids the model in acquiring knowledge of the distinct characteristics, tone, and speech patterns of the targeted individual(Ozbay & Alatas, 2020).

Subsequently, once the model is trained, it is capable of producing new audio content that closely mimics the voice of the individual it was trained on. This generated audio can then be utilized to produce synthetic speech or modify existing audio recordings, creating the illusion that the individual said or recorded something they did not. Within this category, two types of manipulation were identified: text-to-speech and speech-to-speech or voice conversion.

1.5.3. Text-Based

Text-based deepfakes involve generating written content, such as articles, reviews, forum discussions, or social media posts, using deep learning language models (Kumar et al., 2020). These models can mimic human writing style, tone, and vocabulary, resulting in text that convincingly resembles the writing of a specific individual when generated by artificial intelligence.

Text-based deepfakes are divided into synthetic text and artificial intelligence-generated text. Some text-based deepfakes are created and distributed by bots.

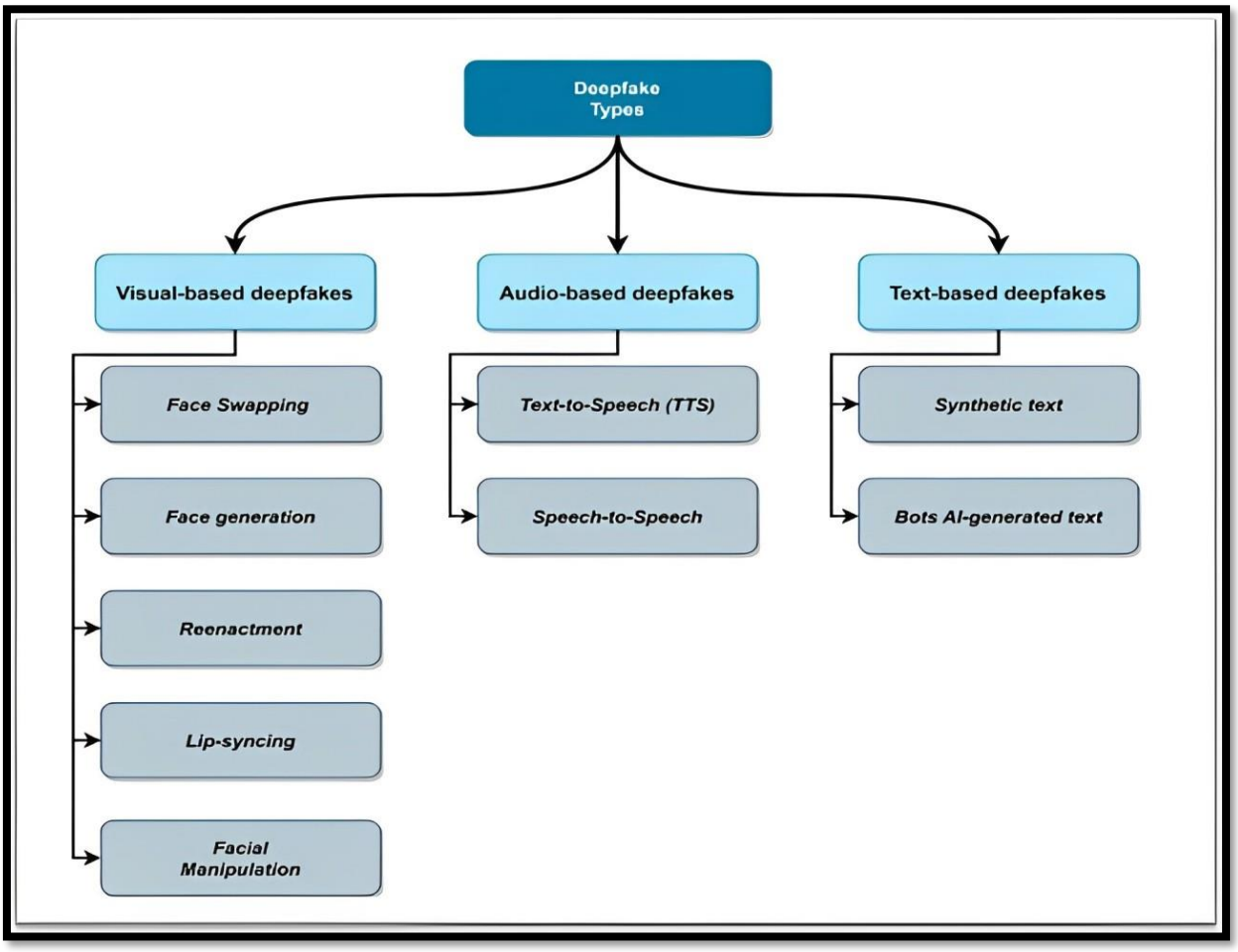


Figure 1.2.*DeepFakes Types.*

1.6. Fake news categories

Fake news can be broadly categorized based on the intent and nature of the false information. Each category has its own implications and impacts on public perception and trust.

1.6.1. Misinformation

Misinformation is defined as information that is inaccurate or misleading. It could spread unintentionally due to honest reporting mistakes or incorrect interpretations. We can also define it as information that may be false, but is not intentionally published (Wardle, 2017).

1.6.2. Disinformation

Disinformation is a false information that is spread deliberately to deceive people or promote biased agenda. (Fetzer, 2004)(Kshetri & Voas, 2017)

1.6.3. Mal-information

The last category of Fake News is defined as a fact-based statement of information but is employed with the intention of causing harm to an individual, a group, or a nation(Wardle & Derakhshan, 2018) .

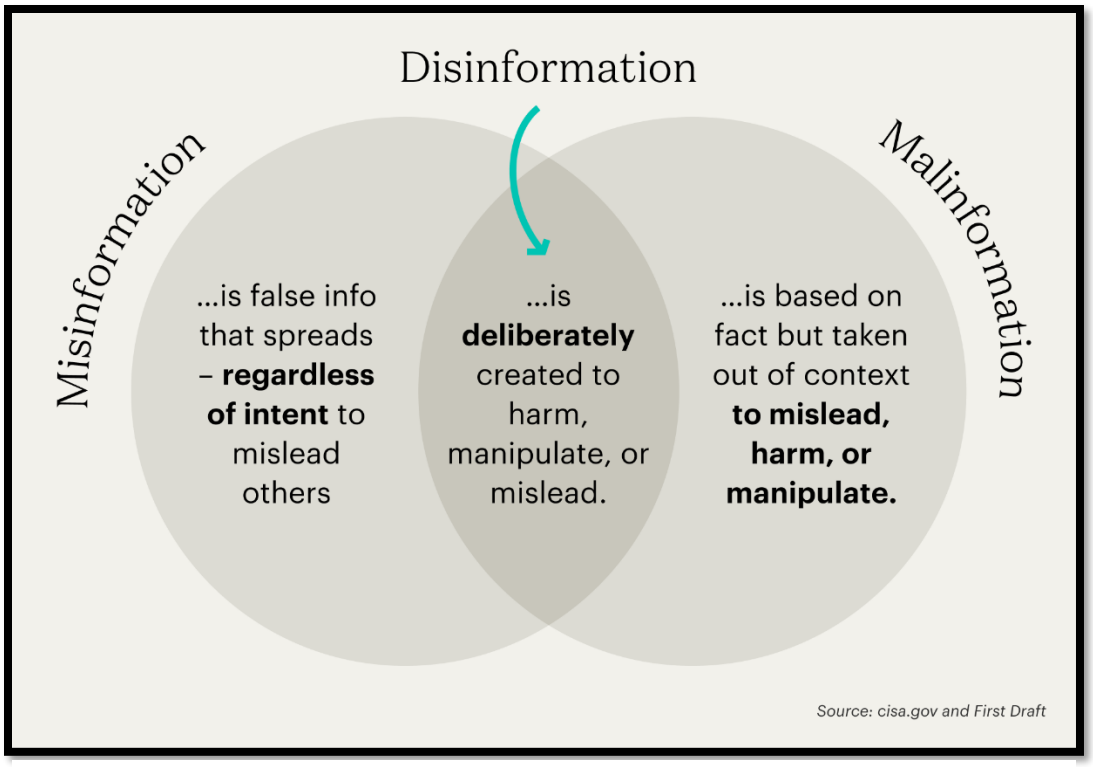


Figure 1.3.*Fake News Categories.*

1.7. Why fake news spread in social media?

Social media has accelerated the dissemination of information, serving as both a rapid conduit for genuine information and a fertile ground for the propagation of fake news (Tambuscio et al., 2015)(Tacchini et al., 2017)(Yang et al., 2018). Several factors contribute to the amplification and spread of misinformation through social media platforms:

- **Easy to share**: Social media platforms are designed to be engaging, allowing users to easily share content and connect with others. This feature makes it easier for information to be manipulated by highly active and partisan individuals, as well as bots, making them powerful sources of misinformation. (Baum & Lazer, 2017).

- **Social bots**: Their role is to automatically produce and spread fake news across social media platforms. Social bots aim to deceive the social media user by spreading rumors, spam, malware, misinformation, slander, and noise (Shu et al., 2017) . Bots are designed to amplify the reach of fake news (Shao et al., 2018) and exploit vulnerabilities that stem from our cognitive and social biases.
- **Social Media Algorithms**: platforms operate as black boxes, the decision they make- from algorithmic curation to human selection and moderation – are not transparent decision (Pasquale, 2015),

Social media algorithms are designed to prioritize content that generate engagement such as likes, shares and comments this can amplify sensational content include the fake news. For example, if the social media algorithm favors posts with provocative or emotionally charged content that tends to elicit strong reactions, this could increase the visibility and spread of misinformation or sensationalism. This dynamic can create a feedback loop, where the algorithm boosts the visibility of content that generates high engagement, regardless of its accuracy or reliability.

1.8. Impact of fake news

Fake news has far-reaching consequences affecting different aspects of society. Understanding these influences is critical to developing strategies to mitigate the spread of false information.

1.8.1. Impact on health

Health information available online is often unreliable (Fu et al., 2016)(Fahy et al., 2014). Many adults obtain health-related information from the internet and social media without considering the credibility of the content. According to (Fox & Duggan, 2013), 60% of US adults search for health information online.

An experimental study conducted by (Waszak et al., 2018) investigated the top health-related fake news stories posted on social media using an app called BuzzSumo. They reviewed eighty of the most shared pages with the goal of detecting the presence of fake news. On each page, they found that 40% of the most shared links contained text classified as fake news.

This type of fake news has a significant impact on the health literacy of individuals, contributing to the spread of medical conspiracy theories and leading to actual behavioral changes (Schwitzer, 2017). The misinformation has been linked to measles outbreaks and deaths, as indicated by the

Centers for Disease Control and Prevention, which reported an alarming trend of measles in the USA by August 2019, exceeding previous records (CDC, 2019).

Additionally, a study on the impact of fake news posts about vaccination on social media revealed that it influenced and changed parents' opinions about vaccination.

1.8.2. Political impact

The impact of fake news on democracies, especially in the context of the 2016 US presidential election and the 2016 UK Brexit referendum.

During the 2016 US presidential election, researchers, including (Allcott & Gentzkow, 2017), speculated on the potential impact of fake news stories spreading on voters. Social media, especially Facebook as a prominent platform, the primary source of news for about 62% of adults in the United States, has played a crucial role in spreading fake news about the candidates, along with the large number of people who believe it, raising concerns about its impact on democratic processes.

Fake news articles are published under names that resemble legitimate news organizations. With several websites coordinating to share and spread fake news stories. The motives behind publishing this news varied, with ideological preferences and financial gain being the primary factors (Townsend, 2016)(Subramanian, 2017). The impact of exposure to fake news during the 2016 US presidential election, compared to television campaign ads, suggests a measurable impact on voters. (Spenkuch & Toniatti, 2016).

In the context of the 2016 Brexit referendum, political bots played a small but strategic role in shaping Twitter conversations (Hern, 2017). Researchers at British universities have conducted extensive studies. These studies indicate that bots on Twitter can influence public opinion, and media reports claimed that 150,000 bot accounts linked to Russia were discovered during Brexit (Payne, 2017).

Overall, studies have highlighted the pervasive influence of fake news and political bots in shaping public opinion during crucial political events, posing challenges to the integrity of democratic systems.

1.8.3. Impact of local rumors and fake news on society

Fake news spread on social media sites is considered a contributing factor to the rise of communal violence and displacement in societies. (Narayanan et al., 2017) delved into numerous cases of public violence in India resulting from the dissemination of rumors. As a consequence, Mohsen Sheikh fell victim to a mob execution in 2014, and the news propagated extensively across social media, contributing to numerous killings and executions in India.

In a study led by (Arun, 2022), it is posited that the spread of rumors via the WhatsApp application presents two distinct problems. The first pertains to misinformation, while the second involves incitement to violence. Chinmayi further explicates how fake news has directly contributed to several incidents of lynching in India, specifically attributing them to content shared on WhatsApp.

1.8.4. Financial impact

Fake news is having a major impact on the financial landscape, as evidenced by a study conducted by University of Baltimore business professor Roberto Cavazos and CHEQ, an artificial intelligence (AI) cybersecurity company. The study reveals that fake news contributes to an annual loss in stock market value of up to $39 billion.

Technological advances and widespread access to information on the Internet have increased the risks associated with fake news for investors. One notable incident was the BlackRock letter hoax earlier this year, in which a fabricated letter attributed to BlackRock CEO Larry Fink caused hundreds of millions of dollars in fluctuations in the asset manager's stock price. Another example concerns a false ABC news report in December 2017, falsely claiming that former National Security Advisor Michael Flynn would testify that President Donald Trump ordered him to contact Russian officials during the 2016 election campaign. This report led to a $341 billion loss for the S&P 500 over 30 years. After the news subsided, the loss decreased to $51 billion.

Looking ahead, researchers estimate at least $200 million in American journalism.[iii]

1.9. Fake News Detection Techniques on social media

Various techniques have been developed to detect fake news on social media platforms. These methods can be broadly classified based on their approach to analyzing the content and context of news.

1.9.1. News Content-Based

Content-based fake news recognition approach attempts to identify fake news by examining either the text or the image inside the news piece (Revez & Corujo, 2021).

1.9.1.1. Knowledge-Based system

It uses a fact checking method in which the supplied claim is checked against information obtained from external sources manually and automatically.

A. manual fact-checking

There are two types of manual fact-checking:

a) Expert-based methods

These methods are based on experts use of export-oriented strategy and rely on human professionals who are educated in certain fields to make judgment to be effective to combat misinformation.

b) Crowdsourced

This method uses the 'wisdom of crowd 'to verify the credibility of news items, platforms like Fiskkit allow individuals to debate news items and determine their veracity. Crowdsourcing fact-checking is less reliable, more difficult to manage, and prone to biases and inconsistent explanations, but offers greater scalability (Khan et al., 2022)(Apuke & Omar, 2021).

B. Automatic Fact-Checking

These methods have been developed to solve problem of manual fact-checking not scaling well the large data, especially those created via usage of social media. These techniques depend on Natural Language Processing (NLP), data mining, machine learning (ML), network/graph theory approaches, and many others, rather than on human brains (I. Ali et al., 2022). There are two stages to the automatic fact-checking process (Zhou & Zafarani, 2020):

a) **Fact extraction: c**oncerned with the collection of facts and the creation of a knowledge base.
b) **Fact-checking:** concerned with determining the authenticity of news articles by comparing them to the information contained in the knowledge base.

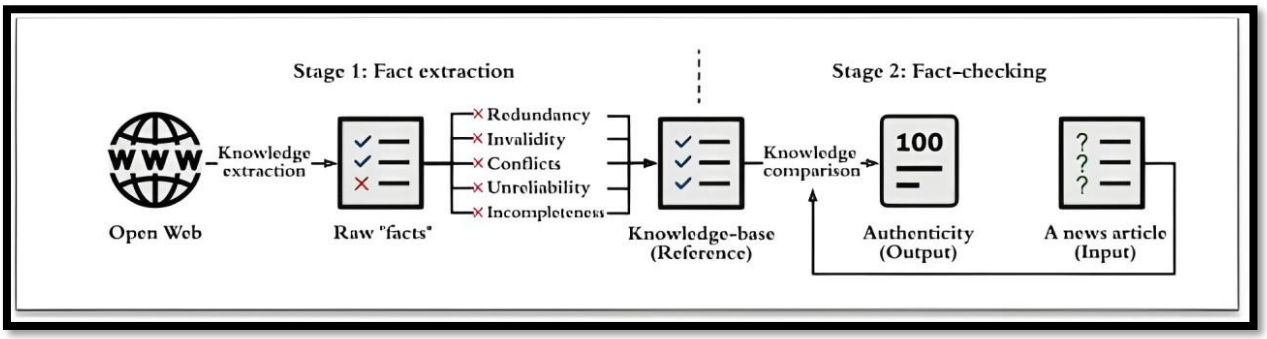


Figure 1.4.*Automatic fact-checking process.*

1.9.1.2. Style-Based

Used to identify fake news and evaluate the writer's goal in deceiving the public, not the legitimacy of the news material itself. In most cases, fake news writers are motivated by a desire to exert influence over large groups of people by spreading inaccurate and misleading information. They typically have their names written in capital letters, with much more proper nouns and fewer stop words to make them attractive. Style-based techniques capture the qualities of writing styles that separate real users from anomalous users (Figueira & Oliveira, 2017).

1.9.1.3. Linguistic-Based

Many researchers, such (Varma et al., 2021), introduced lexical text features and linguistic norms, models such as Social Article Fusion (SAF) and the network computation features model by (Preston et al., 2021). These linguistic analyzes take into account elements such as lexicon, grammar, semantics, and levels of discourse to distinguish between real and fake news. Despite their effectiveness, these strategies face challenges on social media platforms where messages are brief and may lack sufficient linguistic elements for accurate predictions.

1.9.1.4. Visual-Based

he visual approach works on the belief that visual materials can enhance the credibility of news. (Orabi et al., 2020) focused on collecting visual and statistical features to evaluate news credibility through photographic image analysis. This involves evaluating elements such as image content, context and associated metadata to determine the reliability of the visual information presented in the news.

1.9.2. Social Context-Based

Social context-based techniques include user profiling, posting and replies, and network architecture. These factors contribute to understanding how news spreads over time, determining its veracity, and evaluating political positions.

1.9.2.1. Network-Based

Network-based fake news detection examines social relationships and interactions to identify fake news spreaders and understand how misinformation spreads on social media (Oshikawa et al., 2018)(Guimarães et al., 2021)(Khan et al., 2022).

1.9.2.2. Temporal-Based

Temporal studies recognize that online news articles evolve over time, especially when rumors resurface. Researchers such as (Varma et al., 2021) analyzed recurring rumors at the message level across different periods, highlighting patterns of rumors spreading over time.

1.9.2.3. Credibility-Based

Credibility-based methods evaluate the quality and reliability of claims, publishers, or sources to determine trustworthiness.. (Bhavani & Kumar, 2021) propose a method to analyze the credibility of tweets. Real-time assessment of the reliability of tweets is done through web-based systems such as TweetCred.

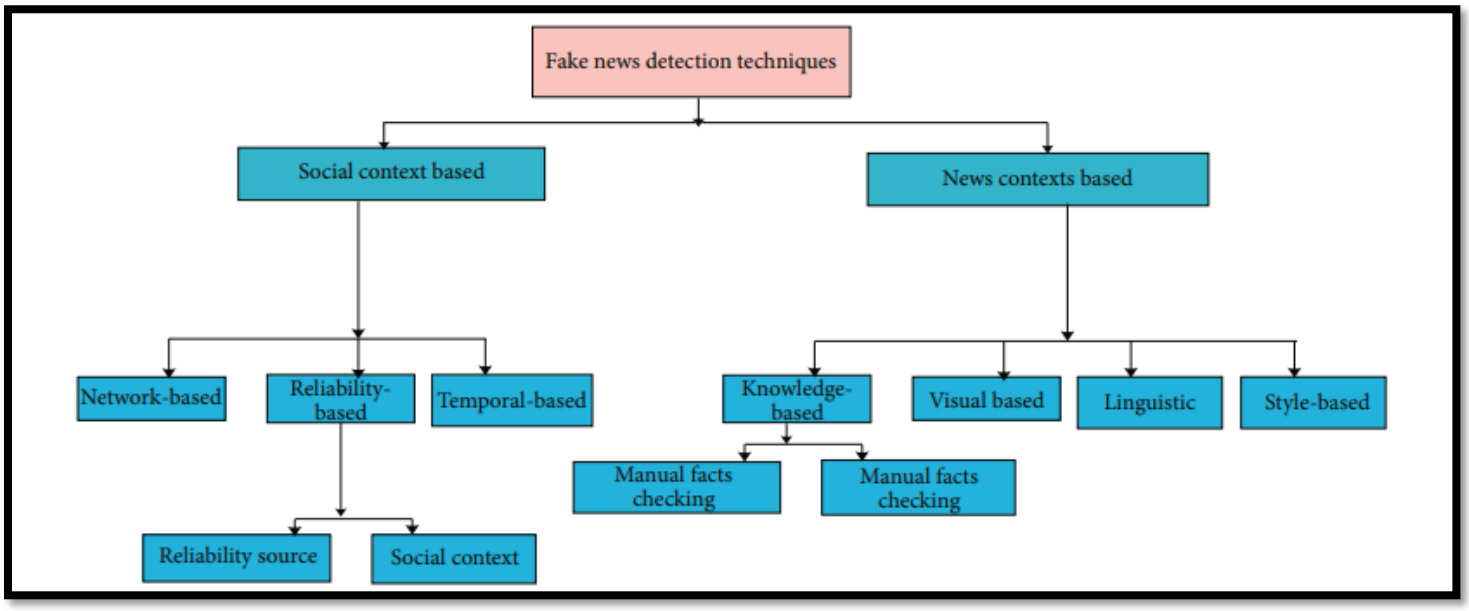


Figure 1.5.*Fake News Detection Techniques on social media.*

1.10. Challenges in fake news detection

Detecting fake news poses many challenges due to the complex nature of misinformation and the dynamics of online platforms.

1.10.1. Variety

This challenge arises due to the diverse nature of fake news concepts, including propaganda, hoaxes, rumors, satire, deep fakes, and various deceptive strategies. Each type of fake news is created with the intention of targeting specific segments of society to achieve particular political, financial goals, or to mislead individuals and influence their opinions. The multitude of fake news categories and the wide range of resources employed in their creation pose a significant obstacle to fake news detection and identifying methods (Zhang & Ghorbani, 2020).

1.10.2. Volume

The viral of social media platforms has facilitated the effortless dissemination and sharing of fake news, accessible to everyone without undergoing any verification procedures. Additionally, the rise of web pages designed to propagate false information, such as denverguardian.com, wtoe5news.com, and ABCnews.com (which often resemble legitimate news organizations), has become prevalent (Allcott & Gentzkow, 2017). These websites are intentionally crafted to spread

hoaxes, propaganda, and disinformation, often driven by financial or political motives. Consequently, the prevalence of such deceptive practices has led to a substantial increase in the volume of fake news circulating within online spaces (Zhang & Ghorbani, 2020).

1.10.3. Velocity

Creators of fake news tend to be short-lived (Allcott & Gentzkow, 2017). For example, many fake news webs pages that were active during the 2016 US election no longer existed after the election campaign. This disappearance of many fake news web pages' post-events is likely an attempt to evade detection. Fake news on social media predominantly focuses on current events to capture attention. The real-time dynamics of online engagement make it challenging to assess the reach and consequences of fake news, eroding trust in authorities and institutions. For instance, online reviews significantly impact customer trust. As fake news continues to affect society, politics, and trust in the online environment, there is an urgent need for an accurate, real-time system to detect and combat deceptive content online (Zhang & Ghorbani, 2020).

1.10.4. Veracity

Refers to the truthful or accuracy of news articles or claim shared on social media. Veracity plays a crucial role in distinguishing between real news and misinformation. However, fake news is deliberately crafted to deceive readers by presenting false information as if it were true. This intentional attempt to mislead complicates the determination of news content's veracity based solely on its surface characteristics (Shu, Wang, et al., 2020).

1.10.5. Valid data

The research community faces challenges in obtaining comprehensive training datasets for fake news detection due to limitations imposed by social media platforms on public data acquisition (Shao et al., 2018). These datasets often lack all necessary information related to fake news. The importance of having a comprehensive dataset to carefully verify the accuracy of news is emphasized (Manish et al., 2022).

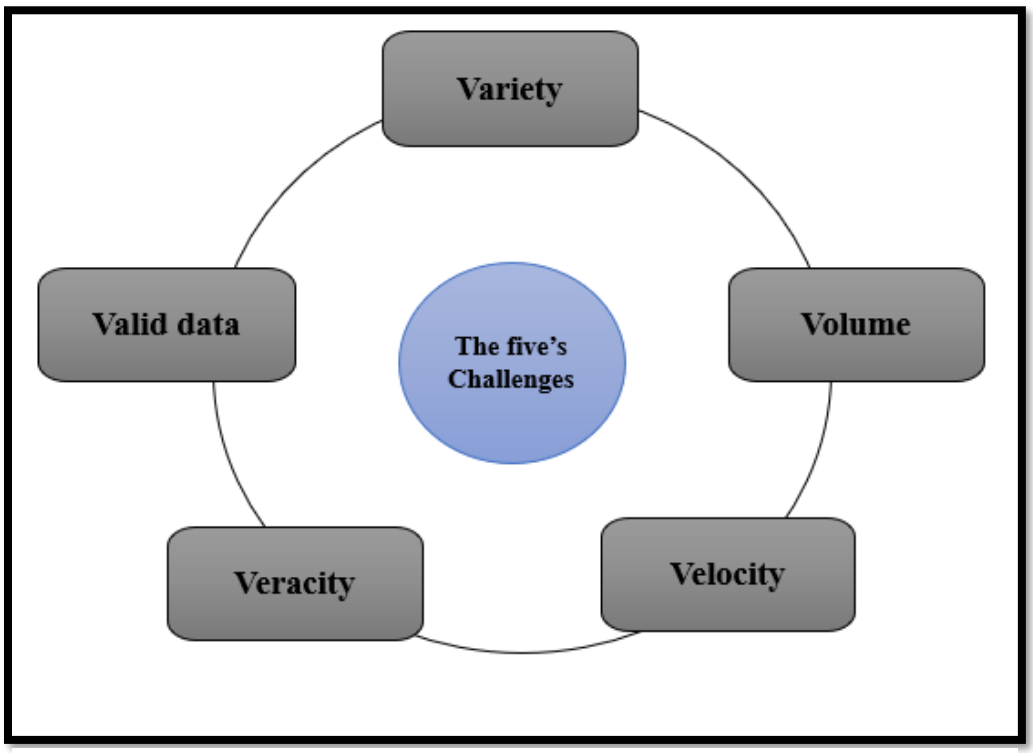


Figure 1.6.*Challenges in fake news detection.*

1.11. Conclusion

In this chapter, we addressed the significant issue of detecting fake news and deepfakes on social media, which have become well known phenomena in this era. We defined the diverse concepts and categories of fake news, exploring the reasons behind its viral nature on social media platforms and its wide-ranging impact on individuals and society, including fields like health. Additionally, we searched into the effective techniques and methods previously used for detection and call attention to the challenges associated with these approaches.

In conclusion, the detection of fake news poses a difficult challenge that requires complicated algorithms and techniques, which will be discussed in the next chapter.

Chapter 2
Literature Review

2.1. Introduction

The spread of fake news in the digital age, especially with the emergence of social media platforms that facilitate the rapid dissemination of information, has led to extensive research efforts to address this challenge. This chapter provides a comprehensive examination of past endeavors and recent developments in detecting and combating fake news and deepfakes. Our focus is on the methodologies and models used, including developments in natural language processing (NLP), deep learning, and machine learning techniques. Through this exploration, we aim to provide insight into the evolving landscape of fake news detection and mitigation strategies.

2.2. NLP

NLP, which stands for Natural Language Processing, also known as computational linguistics, is the subfield of computer science concerned with using computational techniques to learn, understand, and produce human language content (Hirschberg & Manning, 2015). With the proliferation of fake news on social media, the conventional solution to detect fake news, involving professionals such as journalists checking claims against evidence based on previously spoken or written facts, becomes not only impractical but also time-consuming and expensive.

Automated fake news detection using NLP techniques addresses these challenges, solving the budget and time problems while enhancing the accuracy and efficiency of detecting misleading or fabricated information.

The NLP technique consists of data pre-processing and word embedding:

2.2.1. DATA pre-processing

Pre-processing is used to represent complex structures with attributes, binarize attributes, alter discrete attributes and manage lost and obscure characteristics. Multiple techniques are employed during data pre-processing. Due to the fragmented, unstructured and noisy nature of social media data sources, it is necessary to follow a pre-processing approach before feeding the data into the classification model for the purpose of detecting fake news. It is widely acknowledged that during the learning stage, data pre-processing not only saves computational time and space but also helps limit the impact of artifacts and preventing the ingestion of noisy data during the learning process.

The data becomes a logical representation with the most representative and descriptive words after appropriate text pre-processing.

The pre-processing steps are:

A. DATA QUALITY ASSESSMENT:

Data is often sourced from several typically reliable but completely different format. When dealing with a machine learning problem, a significant amount of time is devoted to managing data quality issues. It is unreasonable to expect perfect data, as problems may arise due to human error, flaws in the data collection process, or limitations in measuring equipment. Dataset quality is often a contributing factor to the poor performance of fake news detection models. For this reason, the quality of the data used in any machine learning project will have a significant impact on the chances of success. However, only a few studies prioritize ensuring the quality of the datasets used.

During the disambiguation process, some studies choose to delete rows that do not contain labels, a decision that undoubtedly affects their success in detecting fake news. Duplicate texts are usually removed to ensure the overall quality of the dataset. The data cleansing process is further enhanced by actions such as removing URLs, converting text to lowercase, removing the hashtag character (#), ignoring sign characters (@), and removing numbers. In addition, attention is paid to words containing repeated letters, such as replacing "Likkke" with "Like". Emojis are also processed by replacing positive symbols with a corresponding "positive" word and a "negative" word for negative symbols.

By using these strategies, researchers aim to improve the quality of the dataset, and thus improve the effectiveness of machine learning models in detecting fake news.

B. Train/Validation/Split_test

Train, test, and validation sets can be created from the dataset. The training set is the subset of data that is used to modify the model's weights. The test set is used only to assess the performance of a fully stated model, whereas the validation set is made up of instances used to adjust the model's parameters. When detecting bogus news, data split ratios like 60:20:20, 70:30, and 80:20 are frequently employed. According to the Pareto principle, which states that about 80% of results result from 20% of causes, the 80:20 ratio frequently follows suit. Studies on the identification of fake news frequently use this ratio.

When (Jadhav & Thepade, 2019) examined model performance depending on various data splitting ratios, they discovered that a 75%–25% data split performed better than other models with other splits. While smaller testing data can produce more substantial variation in performance statistics, smaller training data can lead to more variety in model parameter estimates.

To prevent unduly big or tiny differences, investigations must carefully evaluate data splitting. The overall number of cases in each category frequently has a greater impact on the choice than the proportion. Hyperparameters, model architecture, data dimensions, and other factors all play a role in determining the ideal division of test, validation, and train sets. An overview of the benefits and drawbacks of the splitting ratios often employed in the majority of studies is given in Table2-1:

Table 2.1.*The table presents the partitioning of the dataset based on training, validation and testing with advantages and disadvantages*

Tableau 2.1. Train validation and set	advantages	Disadvantages	Reference
80,0,20	- Limited set of validation data or no useful validation data. - When the model contains few or no hyperparameters.	-Lack of validation data can cause the model to overfit.	(Shu, Wang, et al., 2020)

70,0,30	- A sufficient number of test suite is required to evaluate the model.	-With smaller validation data, evaluation metrics such as precision, precision, recall, and F1 score will have large variance and will not load to adjust the model appropriately.	(Verma et al., 2019)
60,20,20	In order to maximize the model's performance, a larger validation set is needed because there are many hyperparameters to adjust.	-With a smaller training set, the model would not have enough data to learn.	(Mahir et al., 2019)

C. Tokenization, Stemming and Lemmatization:

Tokenization is a method of breaking down a text into words. This can be applied to any character. Tokenization is most commonly performed on a space character. Stemming is the process of removing an end to obtain the basic word. Stemming typically includes the elimination of derivational affixes. A derivational affix is one that has one word derived from another. Typically, the derived word belongs to a different class of words than the original.

Lemmatization is a text normalization process that produces the root form of inflated words, analyzes words morphologically, and typically eliminates inflectional endings (Padnekar et al., 2020).

An inflectional ending is a set of letters added to a word's end to change its meaning. The letters s, bat, and bats are a few instances of inflectional ends. Two tests were conducted by (Rusli et al., 2020) to identify bogus news with and without stemming and stop-word removal. To get rid of

every affix and stop word, they employed stemming and stop-word removal. They completed the stemming and stop-word removal procedures, resulting in an F1-score that was 0.82 macro-averaged. Additionally, they were able to attain a 0.8 macro-averaged F1-score without the need for stop-word and stem removal.

Although it took a while to complete the stop-word removal and stemming procedures during the text preprocessing stage, the outcomes were marginally different. A straightforward method for identifying false news via web scraping was given by (Jain & Kasbe, 2018).

They demonstrated how a model's veracity might be verified by routinely updating the dataset through online scraping. Based on text, the writers' accuracy rate was 91%. A little further preprocessing, such as stemming and removing stop words, can significantly enhance the outcome.

2.2.2. Word embedding

A. TF-IDF

Stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text.

Term frequency is defined as the number of times a word appears in a document divided by the total number of words in the document.

$$TF(term) = \frac{\text{the number of times a term appears in a document}}{\text{the total number of items in the document.}}$$

Inverse data frequency defined as the log of the total number of documents divided by the number of documents that contain the word. It is just for standardization.

$$IDF(term) = log(\frac{total\ number\ of\ document}{number\ of\ document\ with\ term\ in\ it})$$

Although this vectorization is successful, the semantic meaning of the word is lost while trying to translate it into numbers (Mandical et al., 2020).

The formula for finding TF-IDF is given as:

$$W_{x,y} = TF_{x,y} * \log\left(\frac{N}{df_x}\right)$$

Where:

$W_{x,y} = Word\ x\ within\ document\ y$

$tf_{x,y} = frequency\ of$ x in y

$df_x = number\ of\ document\ containing\ x$

N = total number of documents

B. Bag of Words (BoW)

It is a text vectorization technique that converts the text into finite length vectors, by counting the occurrences of each word in a document. In addition to data loss, this approach also has limitations. The relative location of the words is overlooked, and contextual information is lost. This loss can be costly at times when measured against the benefit in computing convenience with the ease of use (Thota et al., 2018).

C. GloVe (Global Vectors)

The GloVe method, developed by (Pennington et al., 2014) at Stanford University, distinguishes itself from Word2Vec by leveraging global context to generate word embeddings, giving it a distinct advantage. Unlike Word2Vec, which relies on local context, GloVe focuses on global patterns to generate word embeddings. In GloVe, the semantic relationships between words are derived from the co-occurrence matrix.

In the GloVe methodology, a co-occurrence matrix is generated by iterating through the corpus and calculating the co-occurrence of each word with other words.

Unlike Word2Vec, which uses a predictive model to learn vectors and improve prediction accuracy, GloVe uses a count-based model that takes into account overall word frequency statistics. One notable advantage of GloVe is the ease of training with larger datasets compared to Word2Vec.

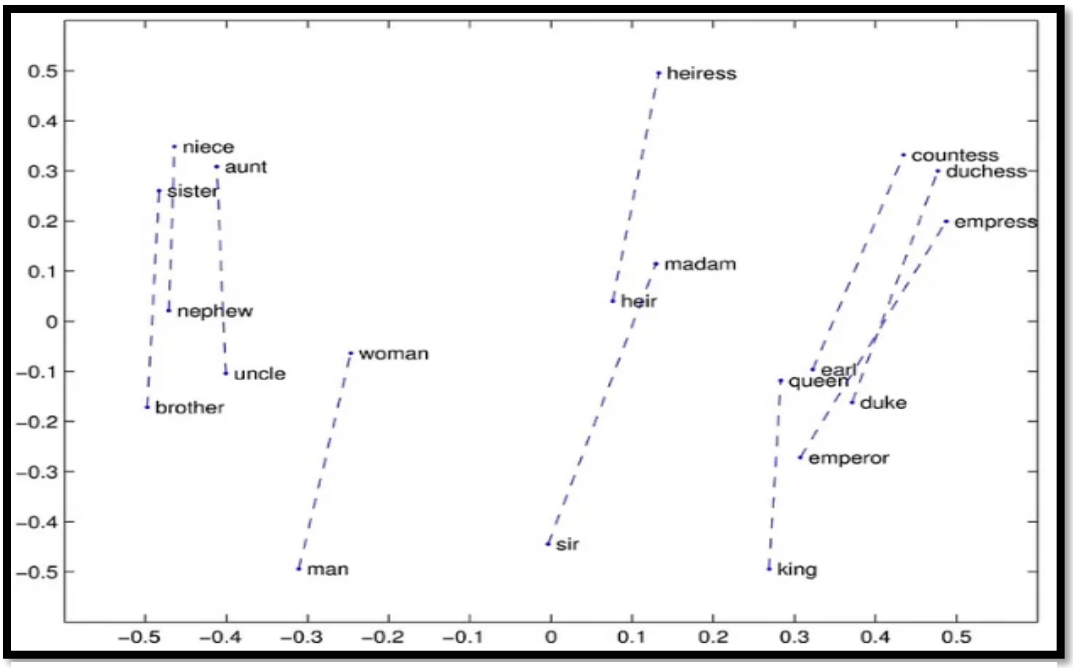


Figure 2.1.*GloVe Architecture.*

D. Word2vec

It is a technique in natural language processing (NLP) designed to derive vector representations for words, capturing information about their meaning and contextual usage within a large corpus.

Word2ec incorporated two architectures such as CBOW and Skip-Gram. In CBOW (Continuous bag of words) the goal is to predict a word by looping many words whereas in Skip-Gram we try to model the contextual words by a given particular word. The below architecture illustrates the same.

In Word2Vec, the similar words are located together in the vector space and arithmetic operations on word vectors can pose semantic relationships.

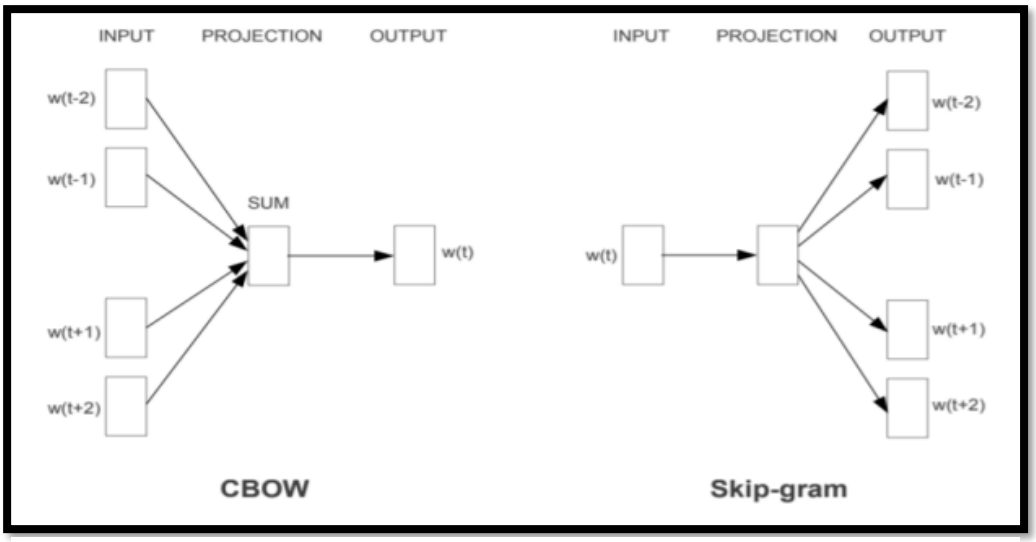


Figure 2.2.*Word2vec architecture.*

E. Doc2vec

Also called a Paragraph Vector a popular technique in Natural Language Processing that enables the representation of documents as vectors. This technique was introduced as an extension to Word2Vec, which is an approach to represent words as numerical vectors. While Word2Vec is used to learn word embeddings, Doc2Vec is used to learn document embeddings.

Doc2Vec has two main variants: Distributed Memory (DM) extends Word2Vec to learn fixed-length vectors for text data by considering context, while Distributed Bag of Words (DBOW) focuses solely on word distribution, making it suitable for structural analysis rather than semantic understanding. Each variant serves distinct purposes in document embedding tasks.

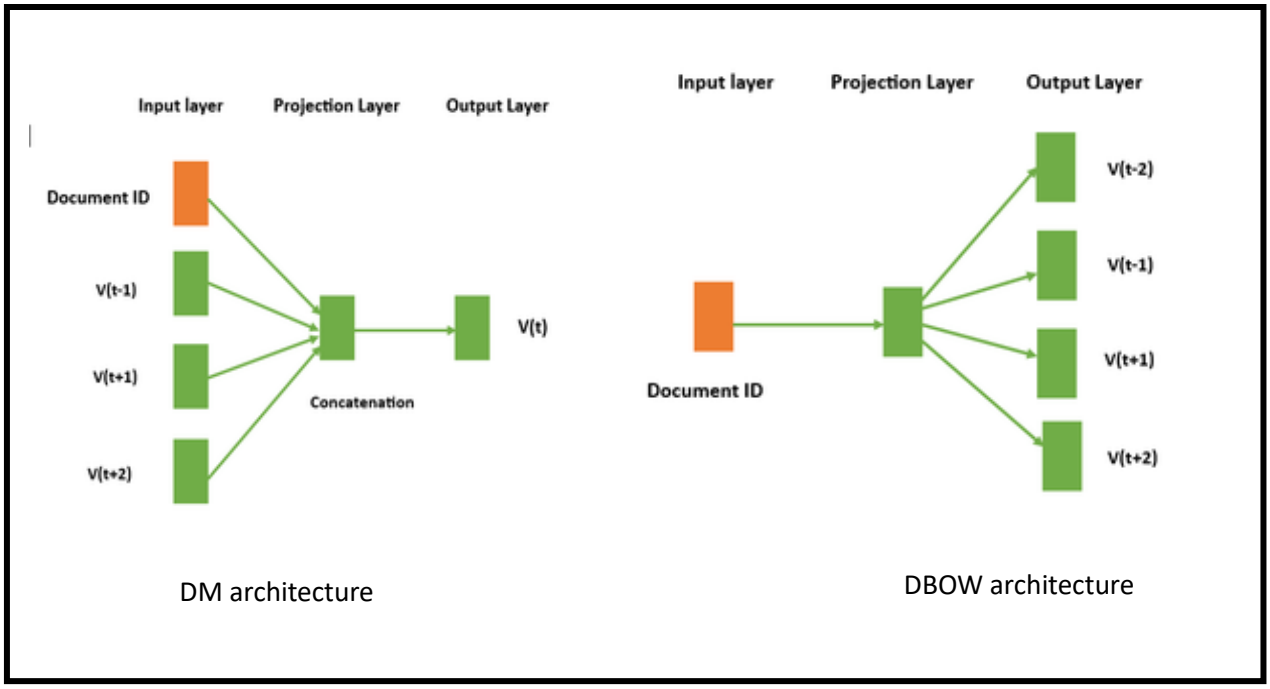


Figure 2.3.*Doc2vec architecture.*

F. Fast-Text

FastText is an open source, free, and lightweight library designed for learning text representations and text classifiers. It is compatible with standard and common devices and allows forms to be efficiently compressed to accommodate publishing on mobile devices.

FastText method distinguishes itself by explicitly incorporating subword information, enabling efficient representation of rare words. It follows the **Skip-gram** model, where each word is depicted as a bag of n-grams or subword units. Each letter **n-gram** is associated with a vector representation, and the final representation of the word is obtained by averaging these vectors.

This subword-aware approach significantly improves the model's performance on grammatical tasks. However, its effect on semantic tasks, such as answering questions about meaning, may not be clear. However, FastText's ability to handle rare words and its lightweight nature make it a versatile choice for various natural language processing applications.

G. BERT

BERT is a pre-trained model that uses the left and right sides of a word to determine its context. BERT announced a new era in NLP because, although precise, it is built on two simple concepts:

pretraining and fine-tuning. BERT is trained on unlabeled data with multiple training challenges in the first step of the model. This is achieved by executing two unattended tasks:

- **Masked ML**

To avoid cycles in which the word being processed could see itself, a deep bidirectional model is trained randomly by covering - masking - some of the input tokens.

- **Predict the next sentence**

Each pre-training set is used 50% of the time in this challenge. When sentence S1 is followed by sentence S2, S2 is classified as IsNext. On the other hand, S2 will be named NotNext if S2 is a random sentence.[iv]

H. GPT_3

GPT is a transformer-based NLP model that can translate, answer questions, compose poetry, solve closes, and execute tasks that require on-the-fly reasoning, such as unscrambling words. The GPT-3 is also used to compose news stories and develop codes, thanks to recent advancements, it is capable of handling statistical interdependence between words. It's been trained on over 175 billion parameters and 45 TB of text gathered from all over the web. It is one of the most comprehensive pre-trained NLP models accessible.

2.3. Machine learning

"Fake news detection using machine learning" refers to the process of identifying and categorizing false information or misleading content that is shared online via social media and other channels by utilizing machine learning algorithms and techniques. This strategy involves training models on labeled datasets, where each data item has a known label indicating whether it is bogus or true news. The trained models can recognize new examples of news items or content as either false or misleading since they have learned the patterns and characteristics typical of fake news. Deepfakes are discovered through the application of machine learning algorithms that distinguish between authentic and altered media content on social media platforms.

The two main machine learning approaches employed are supervised learning (SL) and unsupervised learning (USL). While unsupervised learning concentrates on identifying patterns and structures in unlabeled data without explicit supervision, supervised learning employs labeled datasets to train models. Together, these strategies make it possible to create reliable algorithms for

identifying Deepfakes and fake news in social media, enhancing the accuracy and dependability of online information sources (Chauhan & Singh, 2018).

The types of machine learning include:

2.3.1. Supervised Machine Learning

It is defined as when the model is trained on a "labelled data set". Named data sets contain input and output parameters. In supervised learning algorithms, you learn how to map points between correct inputs and outputs.

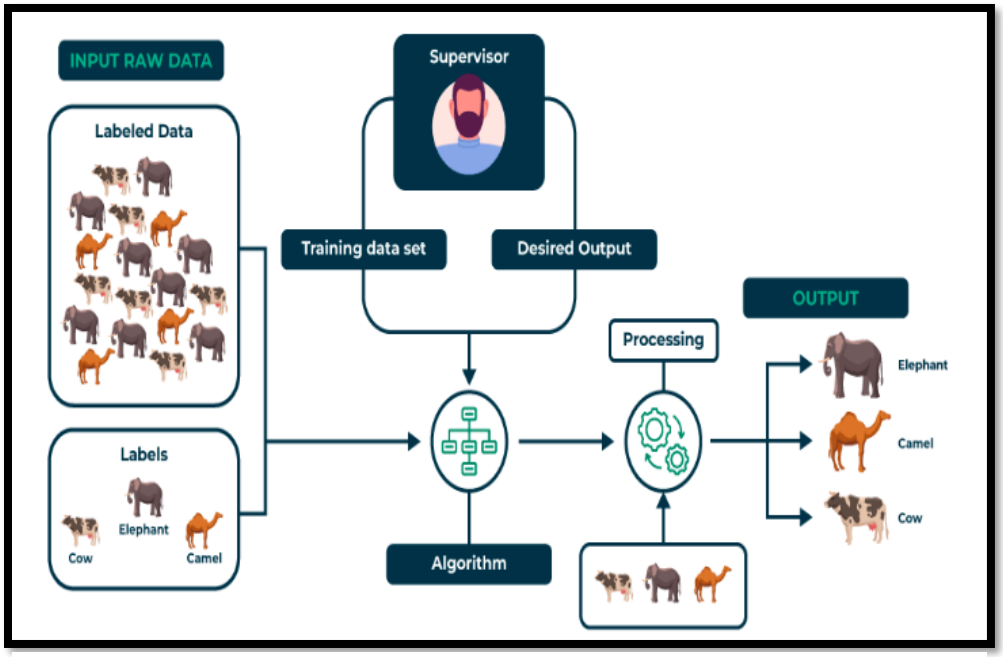


Figure 2.4.*Supervised Machine Learning.*

There are two main categories of supervised learning that are mentioned below:

- **Classification**
- **Regression**

2.3.1.1. Classification

The classification task deals with predicting categorical target variables, which represent discrete classes or labels. For instance, classifying emails as spam or not spam, or predicting whether a patient has a high risk of heart disease. Classification algorithms learn to map the input features to one of the predefined classes.

A. K-nearest neighbour (K-NN):

This technique, which was developed from supervised machine learning algorithms, which were initially applied to handle regression and classification issues in the early 1970s, is distinguished by its simplicity and ease of implementation. The similarity between neighbours is the foundation of this method. Cases are categorized based on the majority of votes cast by their neighbours because the principle of similarity is dependent on the value of K. This happens as a result of related cases being near to one another. (Alkhatib et al., 2013)(Masood et al., 2020)

B. Naive Bayes (NB)

This method is a straightforward probabilistic classification technique that relies on the "naive" assumption of feature independence and the Bayes theorem. It comes from Bayesian theory and is frequently applied to classification challenges in machine learning.

Each data sample in the Naive Bayes classification process is represented by a set of characteristics, and the algorithm determines the likelihood that a particular instance falls into each class based on the attributes that are observed. In fact, Naive Bayes classifiers typically work well despite their simplicity and the erroneous assumption of feature independence, especially when working with big databases or high-dimensional datasets (Masood et al., 2020).

2.3.1.2. Regression

On the other hand, Regression deals with predicting continuous target variables, which represent numerical values. For example, predicting the price of a house based on its size, location, and amenities, or forecasting the sales of a product. Regression algorithms learn to map the input features to a continuous numerical value.

A. Decision tree

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems. It is one of the very powerful algorithms. And it is also used in Random Forest to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning.

2.3.2. Unsupervised Machine Learning

Unsupervised learning is a type of machine learning technique in which an algorithm discovers patterns and relationships using unlabeled data. Unlike supervised learning, unsupervised learning doesn't involve providing the algorithm with labeled target outputs. The primary goal of unsupervised learning is often to discover hidden patterns, similarities, or clusters within the data, which can then be used for various purposes, such as data exploration, visualization, dimensionality reduction, and more.

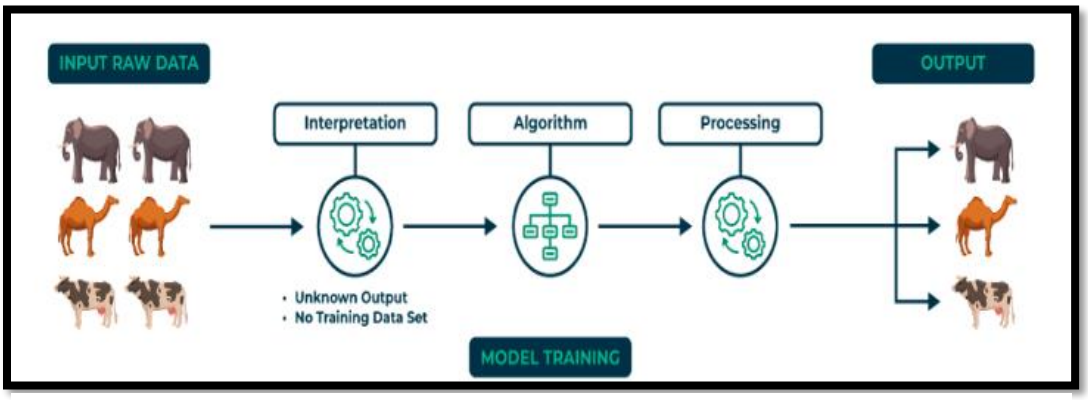


Figure 2.5.*Unsupervised Machine Learning.*

There are two main categories of unsupervised learning that are mentioned below:

- **Clustering**
- **Association**

2.3.2.1. Clustering

It is the process of grouping data points into clusters based on their similarity. This technique is useful for identifying patterns and relationships in data without the need for labeled examples.

A. K-Means Clustering algorithm

K-means clustering is an iterative algorithm designed to minimize within-cluster variance by assigning data points to clusters and updating centroids accordingly. However, its reliance on a predetermined number of clusters, K, poses a challenge, as determining the optimal value is often ambiguous. Moreover, the algorithm's sensitivity to initial centroid placement means different starting configurations may yield disparate outcomes. While K-means is efficient, its effectiveness diminishes when faced with irregularly shaped clusters or varying data densities. Despite these

limitations, K-means remains a widely used clustering method, particularly for applications where computational efficiency is paramount and clusters exhibit relatively uniform shapes and densities.

B. Mean-shift algorithm

This algorithm is a non-parametric, density-based clustering algorithm designed to identify clusters within a data set, especially in cases where the clusters exhibit random shapes and are not clearly separated by linear boundaries.

The basic principle of mean shift clustering involves transforming each data point toward the mode, or peak intensity, of a distribution of nearby points within a specified radius. This iterative process continues until the points converge to the local maximum of the density function, which then represents the clusters in the data. Mean Shift is often referred to as the mode searching algorithm because of this mode searching behavior.

In contrast to the K-Means clustering algorithm, Mean Shift does not require the user to pre-specify the number of clusters. Instead, the algorithm independently determines the optimal number of clusters based on the distribution of the data.

2.3.2.2. Association

Rule-based learning is a technique for discovering relationships between items in a dataset. It identifies rules that indicate the presence of one item implies the presence of another item with a specific probability.

A. Apriori Algorithm

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent item set in a dataset for Boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent item set properties. We apply an iterative approach or level-wise search where k-frequent item sets are used to find k+1 item sets.

To improve the efficiency of level-wise generation of frequent item sets, an important property is used called Apriori property which helps by reducing the search space.

B. FP-growth Algorithm

The FP-growth (Frequent Pattern growth) algorithm is an efficient method for mining frequent patterns in transactional databases. It overcomes the drawbacks of the Apriori algorithm by using a different approach to discover frequent itemsets without the need for candidate generation.

2.3.3. Semi-Supervised Machine Learning

It is a machine learning algorithm that works between the supervised and unsupervised learning so it uses both labelled and unlabelled data. It's particularly useful when obtaining labeled data is costly, time-consuming, or resource-intensive. Semi-supervised learning is chosen when labeled data requires skills and relevant resources in order to train or learn from it.

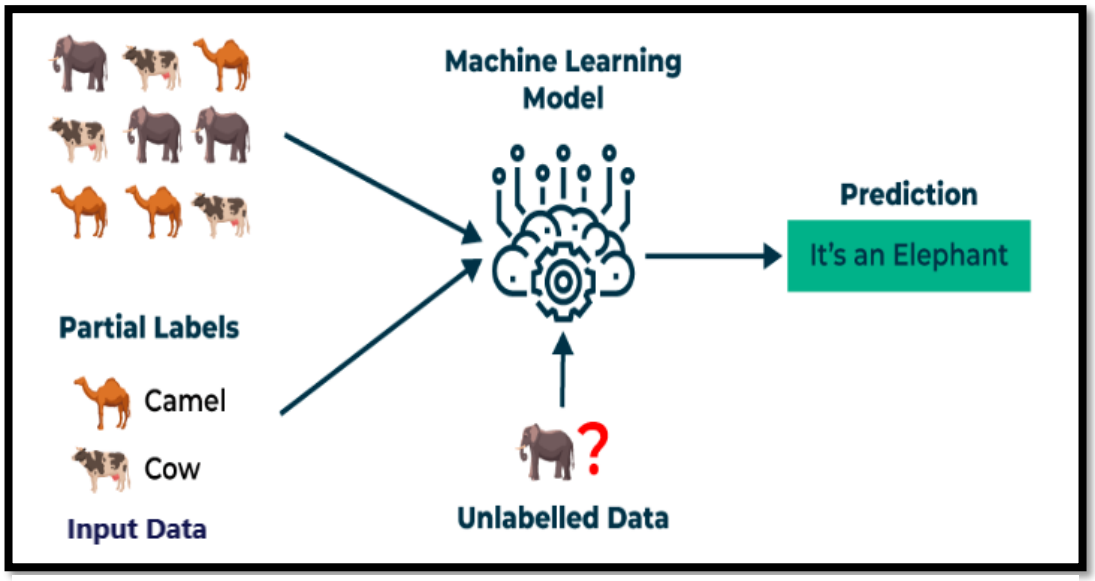


Figure 2.6.*Semi-Supervised Machine Learning.*

A. Graph-based semi-supervised learning

This approach uses a graph to represent the relationships between the data points. The graph is then used to propagate labels from the labeled data points to the unlabeled data points.

B. Label propagation

This approach iteratively propagates labels from the labeled data points to the unlabeled data points, based on the similarities between the data points.[v]

2.4. Deep learning

The detection of fake news has been greatly impacted by deep learning, which uses sophisticated algorithms to go through massive volumes of data and find false information. However, "deepfake" is a term used to describe the use of deep learning algorithms to create realistic-looking but fraudulent content, which frequently raises questions about possible misuse.

2.4.1. Deep Learning Techniques for Fake News Detection

Deep Learning methods are essential for detecting fake news because they use complex algorithms to examine textual data and find patterns that point to false information. Models for natural language processing (NLP), such as transformer models like BERT, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) can be used in these methods. Based on a variety of linguistic signals and contextual information, these models are trained to distinguish between true and misleading information.

2.4.1.1. Neural Networks

An assembly of linked basic processing units, or nodes, that functions similarly to an animal neuron is called a neural network. The inter unit connection strengths, or weights, that the network acquires through adaptation to, or learning from, a set of training patterns, store its processing power. Based on their structure and operation, neural networks can be broadly divided into several types, such as convolutional neural networks (CNNs), feedforward neural networks (FNNs), recurrent neural networks (RNNs), long short-term memory (LSTM) networks, gated recurrent unit (GRU) networks, and more (Gurney, 2018).

The types of neural networks include:

A. Convolutional Neural Network (CNN)

Convolutional neural networks are feed-forward neural networks that process data with a topology resembling a grid and are typically used to evaluate visual images. Another name for it is

a ConvNet. Images are processed using a convolutional neural network for object detection and classification (Yang et al., 2018).

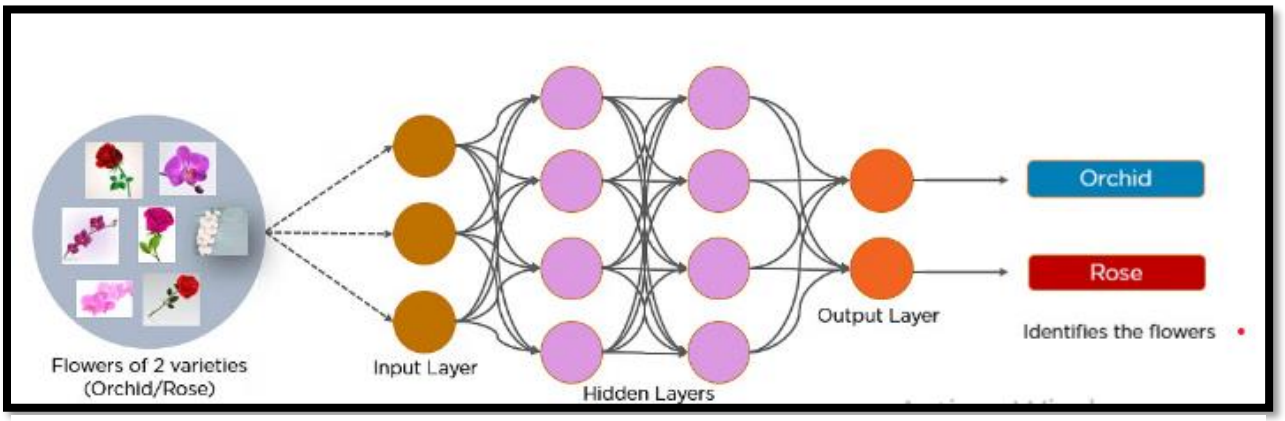


Figure 2.7.*Convolutional Neural Network (CNN).*

B. Recurrent Neural Network (RNN)

RNN is a kind of artificial neural network intended to handle data in a sequential fashion. It is especially useful for problems where the input and output are sequences, such time series prediction, language translation, and speech recognition, because it features connections that allow information to remain (Cheng, 2023).

$$h_t = f_W(h_{t-1}, x_t)$$

new state — old state — input vector at some time step

some function with parameters W

The state consists of a single "hidden" vector h:

ht = fw(ht−1,xt) → ht=f(Wihxt+Whhht−1+bh)

Where:

- ht is the hidden state at time step t
- xt is the input vector at time step t
- Wih and Whh are weight matrices for the input-to-hidden and hidden-to-hidden connections, respectively.
- bh is the bias vector for the hidden state.

- f is the activation function, typically a nonlinear function like the hyperbolic tangent (tanh) or rectified linear unit (ReLU).

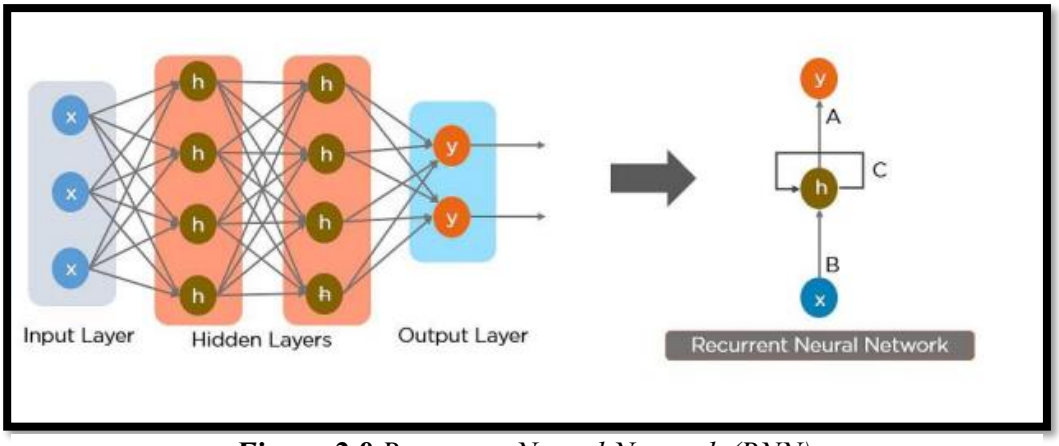


Figure 2.8.*Recurrent Neural Network (RNN).*

C. Feedforward Neural Networks (FNNs)

An artificial neural network that processes data in a forward manner devoid of cycles or feedback loops is called a feedforward neural network (FFNN), also referred to as a feedforward multi-layer perceptron (MLP) neural network. It is among the neural network kinds that are most frequently utilized in deep learning and machine learning (Faris et al., 2016).

Typically, a FFNN has the following layers:

- **The input layer**: is where features or data are received and passed to the subsequent layer.
- **Hidden Layers**: If present, these layers use activation functions to carry out intermediary calculations and transformations on the input data.
- **Output Layer:** Using the calculations made in the hidden layers, this layer generates the final result or forecast.

There are no connections between neurons in the same layer of a FFNN, but every neuron in the adjacent layers is connected to every other neuron in the network. Weights are used to represent the connections between neurons and control the strength of those connections. A summing function and an activation function are present in every neuron in the network.

The weighted sum of inputs plus bias is calculated using the summation function and is expressed by the following formula:

$$S_j = \sum_{i=1}^{n} w_{ij} I_i + \beta_j$$

- Sj is the total input to neuron j
- Wij is the connection weight between input Ii and neuron j
- Bj is the bias weight for neuron j
- n is the total number of inputs to neuron j

An activation function, such as the sigmoid, tanh, or ReLU function, is then applied to the summation function's output to add nonlinearity to the network and generate the neuron's output.

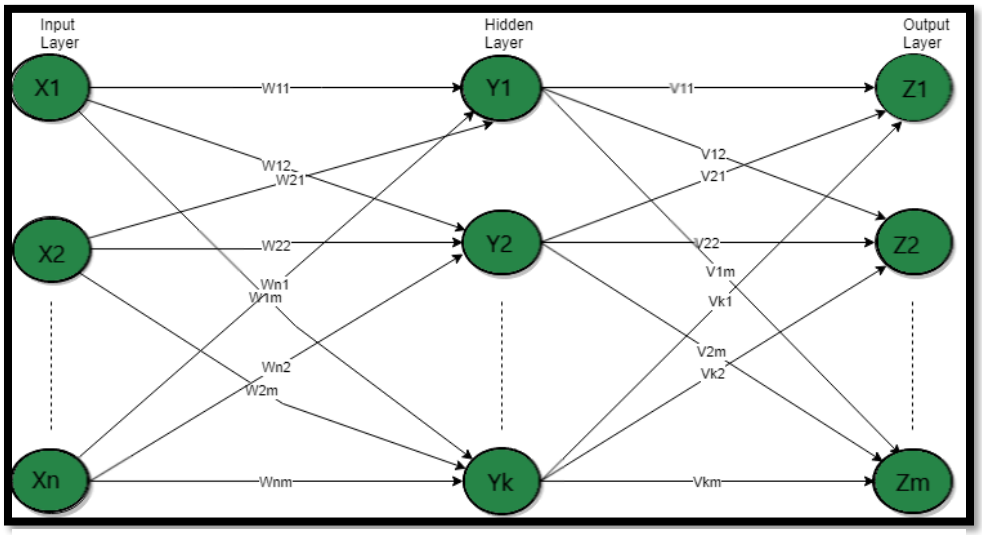


Figure 2.9.*A feedforward network with one hidden layer.*

D. Generative Adversarial Networks (GANs)

Ian Goodfellow first introduced generative adversarial networks (GANs) as a type of generative machine learning model (Goodfellow et al., 2020).

Generative Adversarial Networks (GANs) are trained on training data that closely resembles the original data. Text, images, music, films, visual arts, and photographs are examples of this type of data (S. Ali et al., 2021). Using two neural networks working against each other is the basic concept of GANs:

Generator: The generator network is trained to generate data instances that resemble the desired output, such as real-life photographs. It uses random noise or other input signals to create imaginary data samples.

Discriminator: From the training data, the discriminator network learns to distinguish between instances of true data and instances of false data produced by the generator. It acts as a classifier to determine if data is legitimate.

In a competitive training environment, the discriminator and generator networks are trained concurrently. While the discriminator gains the ability to reliably distinguish between actual and fake data, the generator attempts to produce realistic data in an attempt to trick it. Through this technique, both networks iteratively improve in a feedback loop.

Generating new data instances with underlying patterns and distributions found in the training data is the ultimate purpose of GANs. Applications for GANs can be found in many industries, including robotics, entertainment, healthcare, and the arts. But they also bring up moral questions, particularly in relation to the creation and dissemination of false information (like Deepfakes) and the environmental impact of developing massive AI models (S. Ali et al., 2021).

2.5. Deepfake Generation (deepfake)

Deepfake technology is the process of creating fake content that looks real by utilizing deep learning models, especially generative models like Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs). Large databases of authentic content are used to train these models, which then use those patterns and attributes to create realistic-looking false content. Deepfake technology poses ethical questions because of its potential for misuse, such as producing fake or misleading media, even while it has valid applications in the creative and entertainment industries.

2.6. Related works

The field of natural language processing and machine learning has seen significant progress, with many studies contributing to the development of fake news detection techniques. This section reviews some of the pivotal works in NLP, machine learning, deep learning, and deep fake detection.

One notable method for query expansion, proposed by (Mazari et al., 2013; Mazari & Djeffal, 2022), focuses primarily on improving information retrieval, especially for Arabic texts. While this technique enhances search results by expanding user queries to retrieve more relevant information, it does not directly pertain to machine learning models designed for fake news detection.

2.6.1. NLP Related Works

Recent years have witnessed significant advancements in NLP, mainly due to deep learning methods.

Below are some of the notable works related to Natural Language Processing (NLP) that have contributed significantly to this field:

Word Embeddings Techniques like Word2Vec and GloVe were developed to represent words as numerical vectors, capturing semantic relationships between words. These embeddings are still used in conjunction with newer models. Works like (Ling et al., 2015)(Pennington et al., 2014).

The transformer architecture was proposed by (Vaswani et al., 2017), which became the basis for many modern NLP models due to its scalability and effectiveness in capturing long-term dependencies.

BERT (Bidirectional Encoding Representations from Transformers), a pre-trained language representation model that has achieved state-of-the-art results on a wide range of NLP tasks. Was presented by (Devlin et al., 2018), It has outperformed other models on several NLP benchmarks, including GLUE, SQuAD, and NER.

Then (Liu et al., 2019) proposed RoBERTa, an improved version of BERT that achieved improved performance by expanding the model size, training data, and training duration

(Radford et al., 2019) presented the architecture of GPT, a large-scale unsupervised linguistic model that generates coherent and contextually relevant text, demonstrating the effectiveness of unsupervised pre-training for language comprehension tasks.

Followed by the development of GPT-3, a highly scalable autoregressive language model with 175 billion parameters, which shows impressive learning capabilities in a few snapshots by performing well on various tasks with minimal task-specific training data (Brown et al., 2020).

(Raffel et al., 2020) proposed a unified framework for a wide range of NLP tasks by framing them as text-to-text problems, demonstrating robust performance across different tasks using a single model architecture.

Skip-Gram and CBOW models were created by (Mazari & Djeffal, 2022) by training a word2vec model on a corpus containing 466,424 comments from social media platforms.

2.6.2. Machine learning Related Works

Four machine learning techniques were investigated in the study by (Choudhury & Acharjee, 2023): Support Vector Machine (SVM), Naïve Bayes, Random Forest, and Logistic Regression. Applying SVM and Logistic Regression to the LIAR dataset produced an accuracy of 61%, according to the findings. Furthermore, SVM combined with Random Forest analyzed the Fake Job Posting dataset with an astounding 97% accuracy. Furthermore, SVM by alone showed a remarkable 96% accuracy rate on the News dataset.

The goal of the study "Arabic Fake News Detection Based on Textual Analysis," directed by (Himdi et al., 2022), was to identify fake news in Arabic by textual analysis. Three machine learning techniques were used in the study: Support Vector Machine (SVM), Random Forest (RF), and Naïve Bayes (NB). The dataset featured news articles gathered from Arabic-speaking nations like Saudi Arabia, Egypt, and Jordan on the subject of the Hajj trip to Mecca, Saudi Arabia. It also contained artificially manufactured fake news pieces obtained through crowdsourcing. The findings showed that the accuracy rates for NB, RF, and SVM were 67.6%, 78.6%, and 60.9%, respectively, with F1 scores of 68.4%, 73.5%, and 41.3%.

Notably, using language analysis, the study was able to identify Arabic fake news with over 75% accuracy. Four textual feature categories—emotion, linguistics, polarity, and part of speech—were extracted from real news items that had been altered through crowdsourcing, with linguistic aspects appearing as the most prevalent. With a 78% accuracy rate, the suggested model showed encouraging results, representing a major breakthrough in the field of Arabic fake news detection.

(Aphiwongsophon and Chongstitvatana ,2018) used SVM, neural networks, and Naïve Bayes in a different study to solve the problem of misleading news on social media platforms. They reported remarkable accuracy rates: 90.90% was attained by Neural Networks and SVM, and 96.08% by Naïve Bayes.

Artificial intelligence, according to (Granik and Mesyura ..,2017), offers a practical technique for spotting misleading material. Using the Naïve Bayes classifier, they were able to identify bogus

news from Facebook posts with 74% accuracy. Nonetheless, they acknowledged that more accuracy was still required.

(Mazari & Djeffal, 2021, 2022) created Information Retrieval (IR) test collections using a combination of the pooling strategy on A new Arabic/English test collection was developed, consisting of 600 parallel documents from doctoral dissertation abstracts and 161 queries. the results were validated with the Naïve-Bayes algorithm,achieving a 0.629 F-measure, The study demonstrates that combining machine learning algorithms with pooling strategies can efficiently and effectively build IR test collections.

2.6.3. Deep Learning Related Works:

Amazing neural networks are being developed by a number of academics to detect fake news and deepfake:

The study by (Shu, Wang, et al., 2020) introduces FakeNewsNet, a data repository for researching false information on social media. Along with other sources including BuzzFeedNews, facebookHoax, BS Detector, CREDBANK, BuzzFace, and LIAR, they analyzed PolitiFact and GossipCop datasets using algorithms like SVM, Logistic Regression, Naive Bayes, CNN, and Social Article Fusion. Based on the findings, CNN and PolitiFact were able to attain 0.629 accuracy, 0.807 precision, 0.456 recall, and 0.583 F1 score. CNN obtained a 0.723 accuracy, 0.751 precision, 0.701 recall, and 0.725 F1 score for GossipCop.

(Al-Yahya et al., 2021) a study in which they used several measurements to evaluate different algorithms and models for natural language processing tasks. They employed constant learning rate transformers, transformers with progressive unfreezing and customized learning rate scheduling, and linear and deep learning algorithms. AraBERTv02, AraGPT2, QARiB, ArBert, MarBert, and more models were among them. F1 score, precision, recall, and accuracy were used to gauge performance on datasets such as "ArCOV19-rumors," "AraNews," "Arabic news stance (ANS)," and "Covid-19-fakes evaluation." The F1-Score, Precision, Recall, and Accuracy of AraBERTv02 with customized learning rate and learning rate scheduling were very noteworthy at 0.812187, 0.846519, 0.816502, and 0.825083. With an F1-Score of 0.953345, Precision of 0.956216, Recall of 0.956404, and high performance, QARiB also scored well.

FakeBERT is a potent BERT-based deep convolutional model that was introduced in the publication "FakeBERT: Detecting Fake News in Social Media Using a BERT-Based Deep Learning Approach" by (Kaliyar et al., 2021). It outperforms the most advanced models currently in use, with an astounding 98.90% accuracy in identifying bogus news. For enhanced learning, FakeBERT combines three parallel 1D-CNN blocks with BERT. Subsequent research endeavours to formulate a hybrid methodology that incorporates content, context, and temporal data for binary and multi-class fake news datasets. The study also aims to examine how fake news spreads within echo chambers on social media, considering the opinions of people connected in a graph-like structure.

There are some works that used Bert to detect fake news on social media platforms (Karim et al., 2021)(Liu et al., 2019) proposed DeepHateExplainer architecture, focusing on different BERT models, They used XML, a robust optimized BERT pretraining approach (RoBERTa) to build their models.

(Mazari et al., 2023) conducted a study on multi-aspect hate speech detection, focusing on classifying text into categories such as 'identity hate', 'threat', 'insult', 'obscene', 'toxic', and 'severe toxic'. They employed the BERT model along with Deep Learning algorithms to develop various ensemble learning architectures.

(Rachid et al., 2020) proposed CNN-ATTENTION with AraVec pre-trained word embeddings to classify Cyberbullying Text in Arabic on a dataset of 32,000 deleted comments from Aljazeera.net. These comments are in Modern Standard Arabic (MSA) and various dialects this model achieved 72% of precision.

(Mazari & Djeffal, 2021) uses Deep Learning models (CNN and RNN) applying on corpus collected from social media (Facebook, YouTube and Twitter) about the Hirak_19 (popular protest in Algeria during 2019) written in Algerian Dialect to analyze sentiments and provide a deeper understanding of opinions. The models achieved accuracy of 63.28% (CNN) and 60.97 (RNN) of cross-validation tests.

Several assessments using deep learning models on a new Algerian dialect toxic text detection dataset show that the Bi-GRU model achieved the highest results, with 73.6% accuracy and a 75.8% F1 score. The dataset consists of 14,150 annotated comments labeled as hate speech, offensive language, and cyberbullying (Mazari & Kheddar, 2023).

2.6.4. Deepfakes Related Works

Detecting deepfake texts, or fake texts generated by AI models, is a rapidly developing field of natural language processing (NLP) research. While most research has focused on detecting deepfake images and videos, there are emerging efforts to address the challenge of fake or manipulated text. Here are some related works and approaches:

(Gallé et al., 2021) proposed an unsupervised classifier combined with GPT-2, utilizing features extracted using the n-gram technique on a dataset of 100 books. This model achieves 90% precision and requires a balanced dataset for training, as well as the strictness of exact repetitions.

(Fagni et al., 2021) addressed the identification of deepfake tweets using a fine-tuned RoBERTa-based detector. Their fine-tuned RoBERTa-based detector outperformed other methods, achieving nearly 90% accuracy in detecting deepfake tweets. This model detects short tweets not tested against advanced LLM texts.

(Solaiman et al., 2019) used logistic regression on TF-IDF unigram and bigram features, achieving an accuracy between 93% and 97%.

(Bakhtin et al., 2019) used BiLSTM with GPT_2 and BERT, achieving an accuracy of 97%.

(Nguyen-Son et al., 2017) proposed an SVM model trained on a dataset consisting of 100 English and Finnish books. The model attained an accuracy of 89%.

2.6.5. Datasets

This section discusses several datasets that are commonly utilized in studies aimed at identifying fake news. Collectively, these databases offer a wealth of detailed information, serving as valuable resources for testing and enhancing detection models across various platforms.

- **LIAR**

The LIAR dataset is a freely accessible database for research into fact-checking and identifying fake news. It consists of 12.8 thousand short, hand-filed statements covering a range of settings collected over ten years from POLITIFACT.COM. Editors at POLITIFACT.COM evaluate each statement in this dataset for veracity, providing a trustworthy baseline against which fake news detection algorithms can be trained and evaluated. The usefulness of the fact-checking dataset is increased by the inclusion of comprehensive analytical reports and links to original documents for

each case. The LIAR dataset is an important resource for researchers studying fake news because it is significantly larger than previously accessible public fake news datasets of the same type.[vi]

- **FakeNewsNet**

FakeNewsNet is a dataset compiled from PolitiFact and GossipCop, two well-known fact-checking sites. It contains news articles with explanations from subject matter experts and experienced journalists. The collection also includes social context data related to news stories. FakeNewsNet is a useful tool for training and evaluating fake news detection algorithms because it offers a wide range of data points for analysis and validation due to its combination of tagged news content and social context.[vii]

- **FNC-1**

The primary purpose of the FNC-1 dataset is to aid in stance detection tasks. There are 75,385 tagged headline and article pairs in it. These couples labels specify whether they discuss, agree, disagree, or have no relationship at all. The dataset presents each headline as a statement, offering an organized framework for evaluating and categorizing positions taken in news stories.[viii]

- **Weibo21**

Weibo21 is a benchmark dataset created for multi-domain fake news detection (MFND),[ix] first presented by Nan et al. 4,488 bogus news stories and 4,640 legitimate news items from nine distinct domains are included, along with domain designations. This dataset is a useful tool for testing and creating models and algorithms that are meant to identify false information in a variety of fields (Nan et al., 2021).

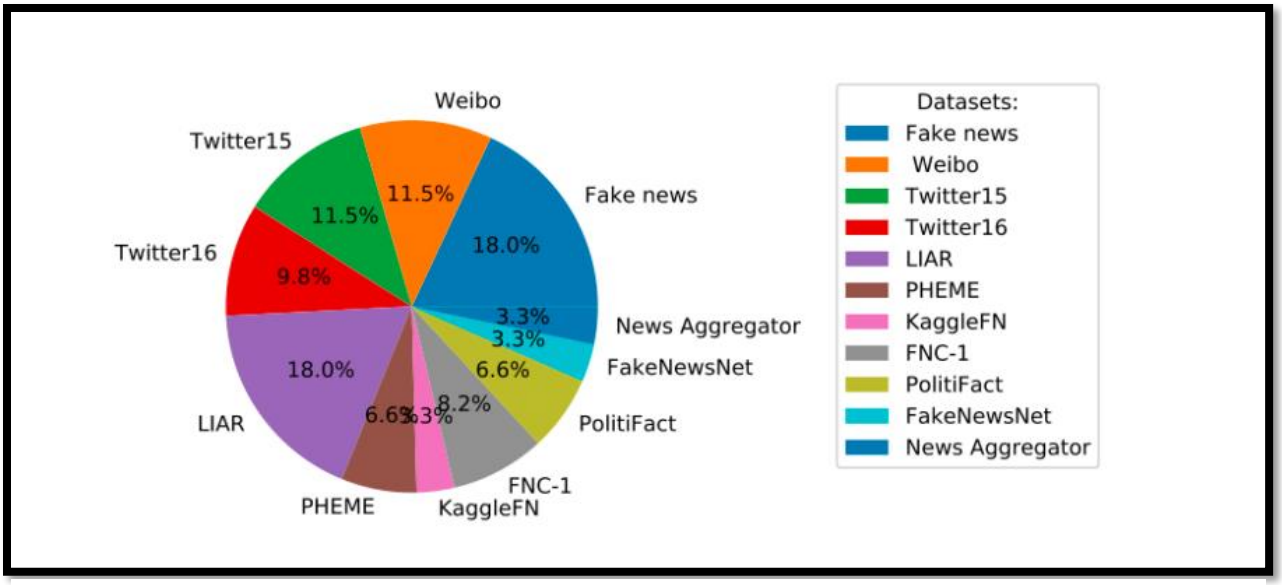


Figure 2.10.*a pie chart showing the benchmark datasets used in studies on the identification of false information.*

Table 2.2.*A brief overview of the published papers focused on fake news detection.*

Article	model	Word_embedding	Dataset	Accuracy	Precesion	Recall	F1
(Kaliyar et al., 2021)	FAKEBERT	BERT	BuzzFeed	0.989			
	CNN	BERT		0.927			
	LSTM	BERT		0.975			
	CNN	Glove		0.915			
(Fouad et al., 2022)	Linear SVM		Real dataset	0.859			0.57
	SGD classifier			0.861			0.845
	CNN			0.780303			

	LSTM			0.838384			
	CNN+LSTM			0.825758			
	BILSTM			0.848283			
	Linear SVM		Benchmark dataset	0.779			0.775
	CNN			0.97679			
	LSTM			0.79865			
	CNN+LSTM			0.727273			
	BILSTM			0.742747			
	CNN+BILSTM			0.7273			
(Harrag & Djahli, 2022)	NLP+CNN		"the arabic fact_cheking and stance detection corpus" from VERIFY ,REUTERS	0.910			0.899
	CNN +RNN			0.7			0.54

(Alyoubi et al., 2023)	MARBERT +CNN		Covid-19 misinformation	0.8707			0.8377
(Ghorbanpour et al., 2023)	FNR-S		public available dataset: twitter	0.79			0.79
	FNR-S		Weibo	0.88			0.88
(Al-Yahya et al., 2021)	AraBERT v02		covid-19-fakes dataset	0.5379	0.9682	0.5426	0.6731
	AraBERT v02		ARA News set	0.800264	0.800255	0.99121	0.203931
	QARIB		ArCov19-Rumors	0.975248	0.956216	0.956404	0.953345
	ArBert			0.958746	0.947729	0.966777	0.953423
(Shu, Mahudeswaran, et al., 2020)	CNN		FakeNewsNet: Polifact	0.629	0.807	0.456	0.583
	SVM			0.580	0.611	0.717	0.659
	LOGISTIC REGRESSION			0.642	0.757	0.543	0.633
	NAIV BAYES			0.617	0.674	0.63	0.651

2.7. Conclusion

This chapter explores the use of NLP, machine learning, and deep learning-based models for fake news and deep fake detection. First, we defined the models and mentioned their most important features, then we reviewed the relevant work in this field and the results obtained. Overall, the use of deep learning and pre-trained models have shown promising results in fake news detection.

Chapter 3
Proposed Approaches

3.1. Introduction

This chapter describes our methods for detecting fake news, employing deep learning, pre-trained and machine learning models. We delve into the design decisions behind the architectures, the preprocessing procedures adopted, and the techniques used to fine-tune models to enhance their effectiveness. Furthermore, we provide various evaluation metrics, including accuracy, F1-score, recall, and precision, enabling a comprehensive evaluation of the performance of our approach.

3.2. Methodology

As shown in the figure3-1, our research focuses on formulating architectures based on machine learning, pre-trained, and deep learning. To produce models equipped with the ability to accurately identify and classify fake news content, thus helping mitigate misinformation. Our models adhere to a systematic series of stages to ensure durability and optimal performance.

The stages of our approach are data collection, text preprocessing, dataset partitioning, word embedding, model building, training, hyperparameter adjustment and evaluation.

First, the dataset is imported in CSV format and goes through preprocessing. During this step, the dataset is cleaned and coded to ensure its quality and consistency. Once pre-processing is complete, the dataset is divided into training, testing, and validation sets.

Then, the word embedding step is performed. Each word in the dataset is converted into a digital representation that captures its semantic meaning. These representations are used as input to the models.

Then the model creation phase takes place, where DL, ML and pre-trained models are created. Next, the hyperparameter adjustment is performed to tune once, and the training process begins with the training set. The model is trained on the input data, and an iterative optimization process occurs to update the model weights. The parameters depend on the patterns and features of the training data.

Finally, once the model has been trained on the training data and has generated classifications for the testing, evaluation is performed to assess the performance of the model.

This evaluation helps in understanding how well the model generalizes to new, unseen data and provides insights into its accuracy, precision, recall, F1-score, and other relevant metrics.

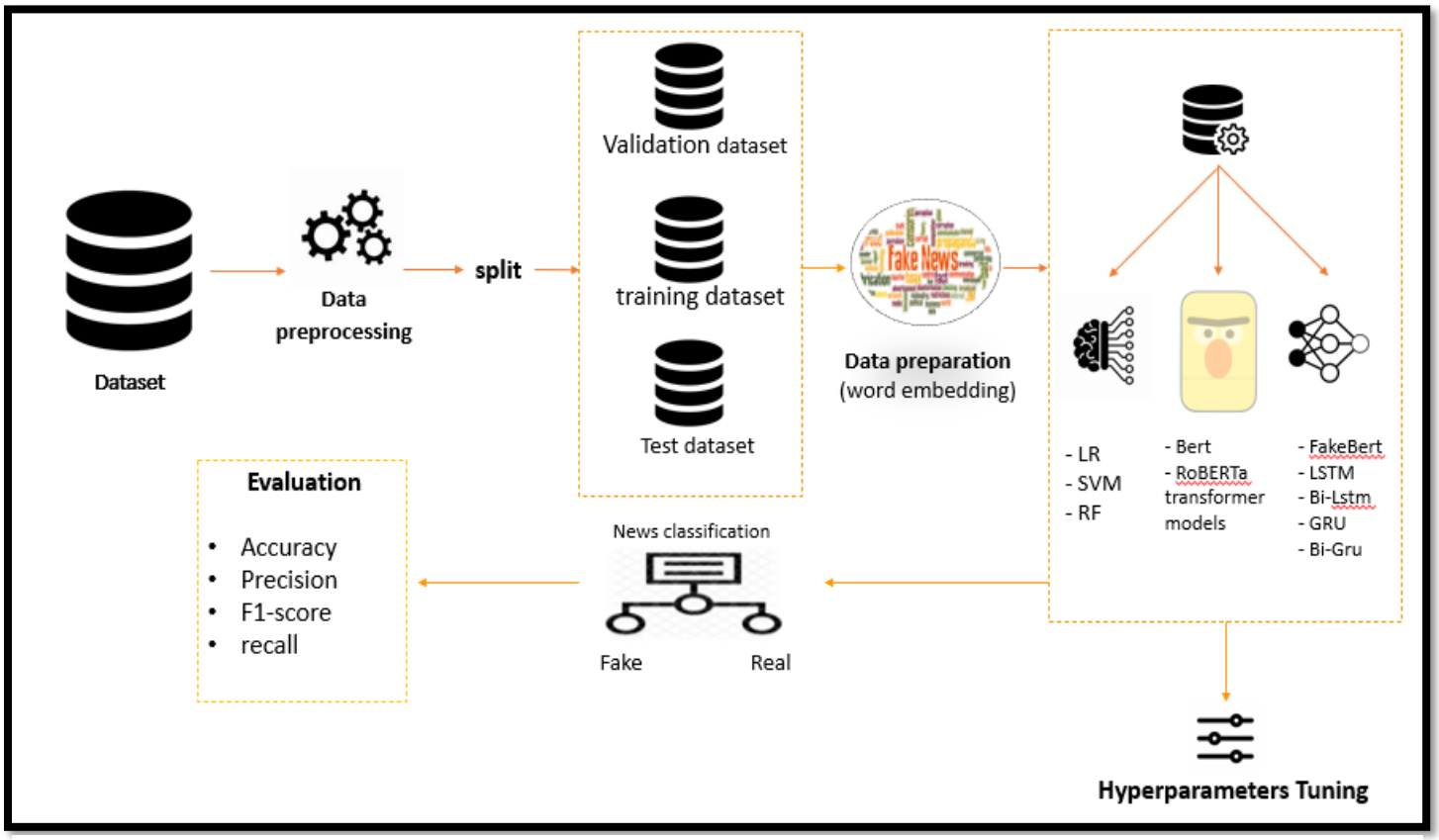


Figure 3.1.*Proposed Approach Overview flowchart.*

3.3. Proposed Approach

In recent years, several advanced neural network architectures have been developed to address challenges in natural language processing, especially in the field of fake news detection. These methods take advantage of the strengths of different models to improve accuracy and efficiency.

3.3.1. FakeBERT

A deep learning approach based on RoBERTa (a variant of BERT). It combines different parallel blocks of a single-layer deep Convolutional Neural Network (CNN) with RoBERTa. This combination is useful for handling ambiguity, which is a significant challenge in natural language understanding. Specifically, FakeBERT aims to improve fake news detection by leveraging both RoBERTa's contextual embeddings and the features extracted by the CNN. The architecture is

designed to capture semantic information and long-distance dependencies in sentences, enhancing its ability to classify news articles as real or fake.

3.3.1. LSTM

A variant of an RNN network, as proposed by (Hochreiter & Schmidhuber, 1997). The LSTM network resolves the challenge of long-term dependencies inherent in classical RNNs by use multiple gates and an internal memory cell to control the flow of information.

3.3.3. GRU

Gated Recurrent Units (GRUs), introduced by (Chung et al., 2014), represent a streamlined alternative to traditional LSTM networks. While similar to LSTMs in their ability to handle variable-length input sequences and retain past information, GRUs employ a single update gate to decide which information to retain and a reset gate to determine what to forget. This simplified architecture makes GRUs easier to train compared to LSTMs, yet they often deliver comparable performance across various tasks (Yang, S. et al.2020).

3.3.4. Bi-LSTM

Bi-LSTM, an extension of Bidirectional RNNs (BRNNs) introduced by(Schuster & Paliwal, 1997)(M. Schuster and K. K. Paliwal, 1997), utilizing information from both past and future contexts by splitting state neurons into positive and negative time directions. By training the network bidirectionally, Bi-LSTMs adeptly capture dependencies in sequential data. Several studies, including (Singh et al., 2021), have demonstrated the effectiveness of Bi-LSTMs in tasks like time series prediction.

3.3.5. Bi-GRU

The bidirectional GRU (Bi-GRU), presented by (Bhuvaneswari et al., 2019), is a type of RNN that analyzes sequential data in both forward and backward directions, leveraging information from both past and future contexts for predictions or classifications. Bi-GRU, composed of two stacked layers of GRU cells, processes data simultaneously in both directions. Widely employed in natural language processing tasks such as language translation, sentiment analysis, and text classification, Bi-GRU proves particularly effective where current context depends on both past and future events.

3.3.6. BERT

A stack of Transformer encoder layer (Vaswani et al., 2017) that consist of multiple self-attention ‘‘heads’’ to compute key, value, and query vectors for each token in a sequence. These vectors contribute to a weighted representation, aggregated and processed through fully connected layers. The workflow typically involves pre-training, focusing on tasks like masked language modeling (MLM) and next sentence prediction (NSP), followed by fine-tuning for specific applications. Input representation involves tokenizing words into wordpieces (Yonghui Wu et al.., 2016) and combining embeddings for tokens, positions, and segments, with [CLS] tokens used for classification and [SEP] tokens for segment separation. Various BERT versions, like "base" and "large," vary in heads, layers, and hidden state size.

3.3.7. RoBERTa

A natural language processing model that is constructed on top of BERT in order to improve its performance and overcome some of its flaws. RoBERTa was created as a consequence of collaboration between Facebook AI and the University of Washington.

3.3.7. Logistic regression (LR)

A supervised machine learning algorithm which has the advantage of yielding a probability model that can be useful in many applications. according to (Hoi SCH, Jin R, 2006) (LR) is a discriminative model that can be used for probabilistic categorization. It outputs the posterior probabilities for test examples that can be conveniently engaged in other systems. If our ultimate goal is classification, then given a test example x, LR can directly estimate the conditional probability of assigning a class label y:

$$P(y|x) = \frac{1}{1 + \exp(-y\alpha^T x)}$$

3.3.8. Random Forest

This is regarded as one of the primary classifiers in supervised machine learning since it is composed of a set of decision tree classifiers, each of which votes for the most well-liked input vector classification category after randomly selecting samples for each decision tree(Pal, 2005)

3.3.9. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm mainly used for classification tasks. Its main goal is to find the optimal hyperplane in an N-dimensional space that effectively separates data points belonging to different classes. The hyperplane aims to maximize the margin between the closest points of different classes, providing robust classification boundaries.

SVMs can be classified into two main types based on the nature of the decision boundaries:

- **Linear SVM:**

Linear SVMs use linear decision boundaries, such as a straight line in 2D or a hyperplane in higher dimensions, to separate data points from different classes, finely separating the data by a straight line or hyperplane, which increases the margin between classes.

- **Nonlinear SVM:**

Nonlinear SVMs are used when data cannot be separated linearly, requiring nonlinear decision boundaries. Kernel functions are used to transform the original input data into a higher dimensional feature space, where linear separation is possible.

By defining nonlinear decision boundaries in this transformed space, nonlinear SVMs can effectively classify separable nonlinear data.

3.3.10. DeepFake Detection Approach

With the increasing integration of AI-generated content into digital communications, especially on social media, distinguishing between human and AI-generated texts has become a significant challenge. our aproache investigates the effectiveness of Gemini model in this domain, utilizing various metrics and scenarios to assess their accuracy and reliability.

3.4. Metrics Utilized for Evaluating Model Performance

Evaluation metrics serve as the compass guiding learning-based models toward accuracy and efficiency. According to (Blackledge & Atapour-Abarghouei, 2021)(Jiang et al., 2021) we utilize the metrics commonly used in previous studies and contributions regarding fake news to assess the effectiveness of the models.[x]

3.4.1. Confusion matrix

A confusion matrix is a summary of correct and incorrect predictions and helps visualize the outcomes.

Confusion matrix something looks like this:

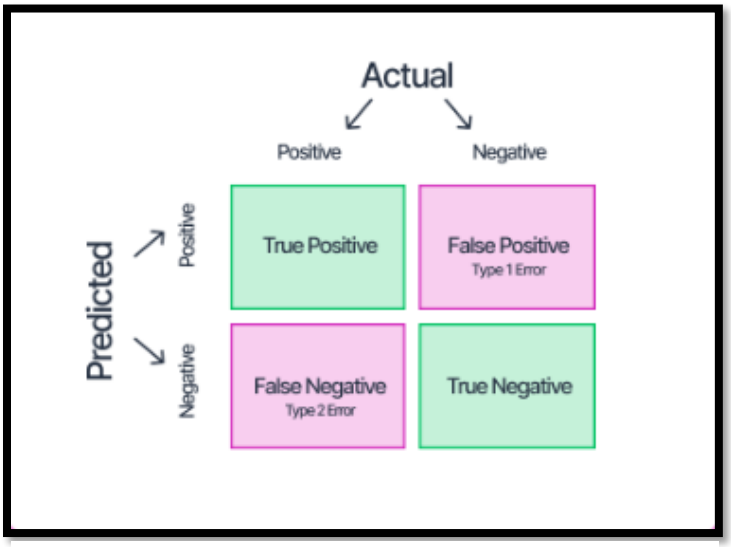


Figure 3.2.*Confusion matrix.*

where,

- **True Positive (TP)**: Predicted positive and it's true.
- **True Negative (TN):** Predicted negative and it's true.
- **False Positive (FP):** Predicted positive and it's false.
- **False Negative (FN):** Predicted negative and it's false.

A. Accuracy

Accuracy is a metric that provides an overall assessment of a model's performance across all classes. It is useful when all classes are of equal importance. It is computed by dividing the number of correct predictions by the total number of predictions made by the model.

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

B. Precision

A performance metric that evaluates the accuracy of positive predictions made by a model. It measures the proportion of correctly predicted positive instances out of all instances predicted as positive. It is calculated as the number of true positive predictions divided by the number of true positive and false positive predictions.

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

C. F1-Score

A metric that represents the harmonic mean of precision and recall. It provides a single value within the range [0,1], indicating the balance between precision (correctly classified instances) and recall (missed instances). It serves as a measure of how accurate and robust our classifier is.

$$\text{F1} - \text{Score} = \frac{(2 * \text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

D. Recall

Also known as sensitivity or true positive rate. It is the ratio of the number of true positive predictions to the total number of actual positive instances in the dataset. Recall measures the ability of a model to identify all relevant instances.

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

3.5. Conclusion

In this chapter, we have presented a comprehensive approach for detecting fake news using deep learning, machine learning, and pre-trained models. The proposed methodology encompasses various steps, including data preprocessing, tokenization, sentence encoding, and the utilization of models such as FakeBert, LSTM, Bi-LSTM, GRU, Bi-GRU, BERT, RoBERTa, LR, SVM, RF , and Gemma API.

To evaluate the effectiveness of the models, we have suggested metrics such as Accuracy, Precision, Recall, and F1-score. In the upcoming chapter, we will implement the suggested models to enable a comparative analysis and determine the most effective one.

Chapter 4
Implemented Approaches

4.1. Introduction

In this chapter, we explore into the execution and evaluation of the deep learning, pre-trained and machine learning models proposed in Chapter 3 for fake news detection. We discuss the development environment, coding framework, dataset selection, preprocessing steps, and the training, validation, and testing of our architectures.

4.2. Data Used for Training and Test

In this section, we present the datasets utilized in our study.

4.2.1. ISOT Fake News Dataset[1]

Fake news dataset created by Ahmed (Ahmed et al., 2017) (Ahmed et al., 2018) at the University of Victoria, includes a collection of articles drawn from real-world platforms. The real news articles were obtained by crawling Reuters.com, a reputable news website, while the fake news articles were obtained from various unreliable websites reported by organizations such as Politifact and Wikipedia. Although the dataset covers diverse topics, a large portion of articles revolve around political issues and global news.

The dataset is organized into two CSV files. The first file, "True.csv," contains more than 12,600 articles retrieved from Reuters.com, while the second file, "Fake.csv," contains more than 12,600 articles from various fake news outlets. Each article entry includes details such as title, full text, genre (real or fake), and publication date. To ensure consistency with data collected for Kaggle.com, the focus was primarily on articles published between 2016 and 2017. While the collected data underwent cleaning operations, punctuation and errors found in fake news articles within the text were preserved.

[1] https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/
Downloaded: 04/03/2024

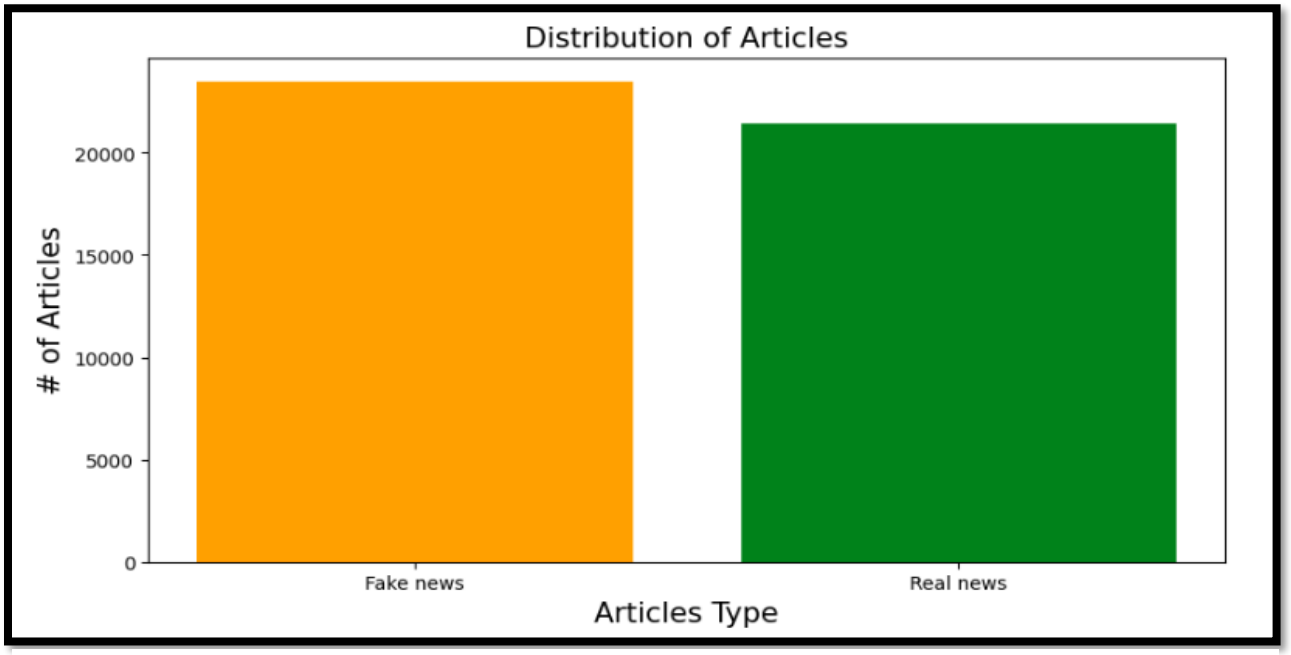


Figure 4.1.*ISOT Fake News Dataset.*

4.2.2. AFND[2]

A large and diverse Arabic fake news dataset created by (Khalil et al., 2022) by the support of Zayed University Research office. This dataset is collected from 134 public and free news website. The identity (Uniform Resource Lector URL) of each website is replaced with source1, source2, source3…etc, in the JSON objects in a file named "sources.json" under the main directory to anonymize the news source websites. Each object has an anonymous name of the news source and a label for the news source (credible, not credible, or undecided).

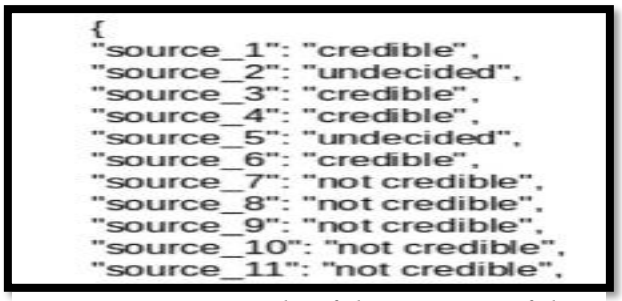


Figure 4.2.*Example of the contents of the JSON file named "sources.json".*

[2] https://data.mendeley.com/datasets/67mhx6hhzd/1/ Downloaded: 01/05/2024

The scraped public Arabic articles are stored in 134 sub-directories within the directory named "Dataset". Each sub-directory is named according to the anonymous names (e.g. "source_1") of the 134 news sources. Each sub-directory has a file named "scraped_articles.json" which contains the information of the articles that were scraped from the public news source and stored as an array of JSON objects. Each object stores the title, text, and the publication date of the article.

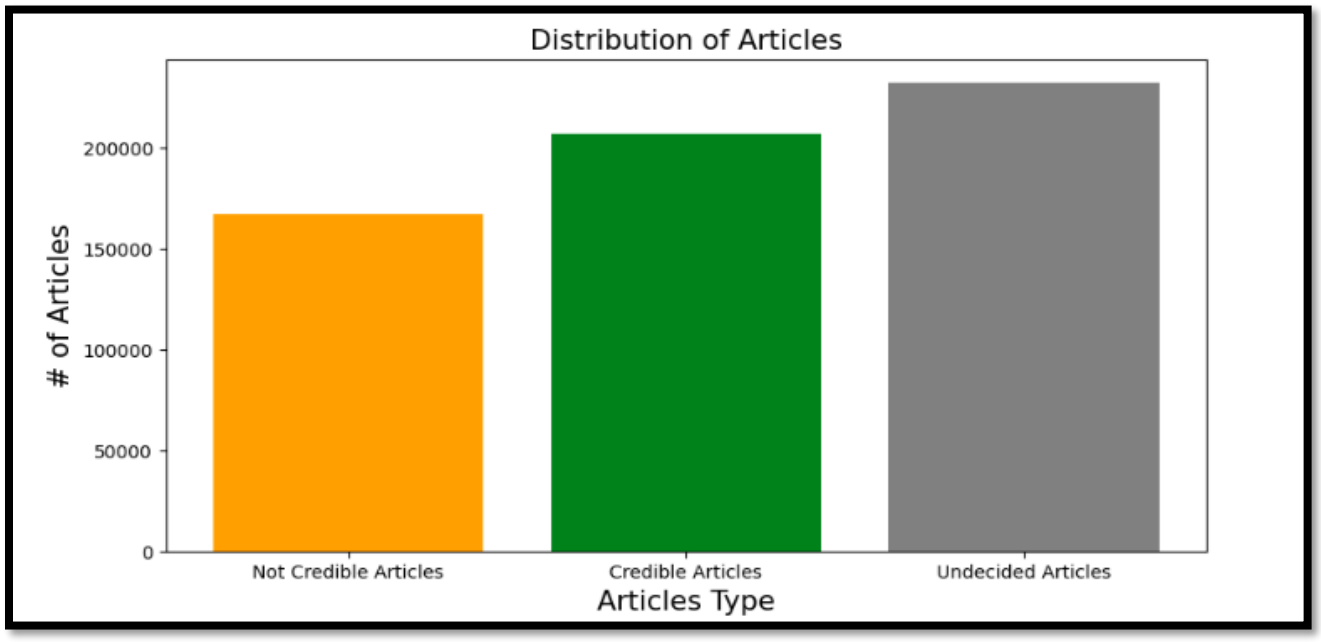


Figure 4.3.*AFND Arabic fake news dataset.*

4.2.3. TweepFake Dataset [3]

TweepFake is a dataset specifically designed for detecting machine-generated content in the form of tweets. It focuses on identifying deepfake tweets, which are generated by AI models rather than real human users.

The dataset contains both real deepfake tweets (posted on Twitter) and genuine tweets from human users. This combination allows researchers to train and evaluate deepfake detection models effectively.

TweepFake includes tweets from a total of 23 bots, which mimic 17 human accounts. These bots use different generation techniques, including Markov Chains, RNN, RNN+Markov, LSTM, and

[3] https://www.kaggle.com/datasets/mtesconi/twitter-deep-fake-text/ Downloaded: 22/05/2024

GPT-2. To ensure balance, the dataset also includes randomly selected tweets from humans that the bots have imitated.

Overall, TweepFake has 25,572 tweets, half of which were created by humans and half by bots.

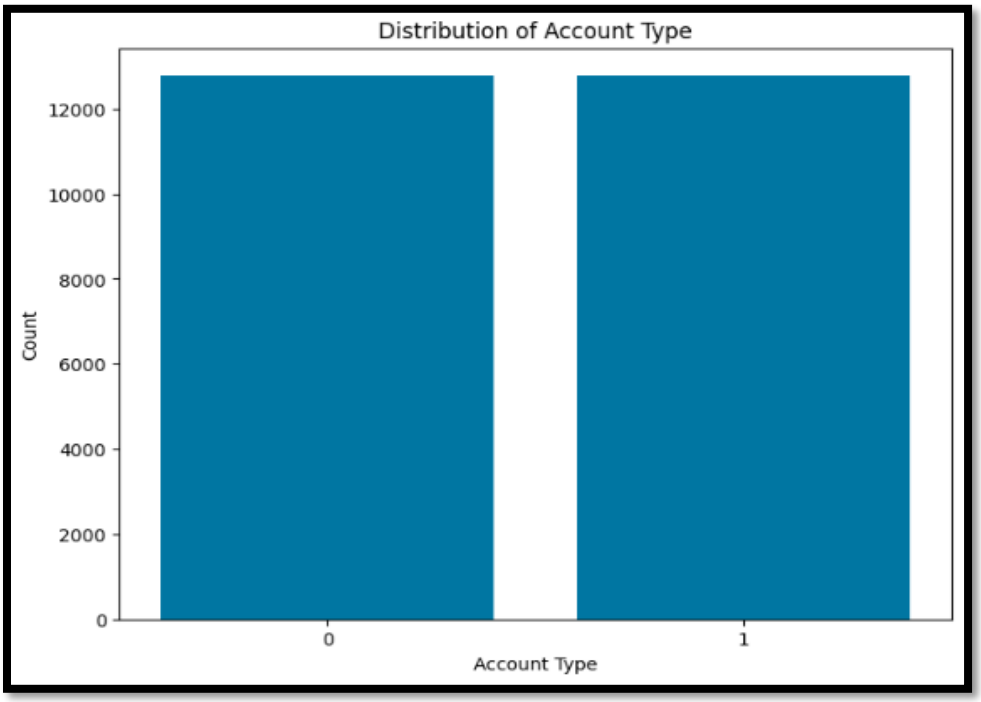


Figure 4.4. *TweepFake Dataset.*

Due to the limited capabilities of our equipment, we were unable to use all datasets in their entirety. We used 100,000 randomly selected rows from AFND and 18,000 rows from TeepFake, while we were able to use the full ISOT dataset.

4.3. Environment and Development Tools

4.3.1. Python[4]

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

[4] https://www.python.org/

Python supports modules and packages, which encourage program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

In the Kaggle notebook execution environment, Python version 3.10.13 is utilized, alongside specific modules mentioned here:

4.3.2. Libraries

A. For Data Manipulation

- **Pandas**[5]: is an open-source, BSD-licensed library that offers high-performance data structures and data analysis tools for the Python programming language. It simplifies the handling of structured data, including CSV and Excel files, by providing functionalities for reading, manipulating, and analyzing data frames effortlessly.
- **NumPy**[6]: is the fundamental package for scientific computing in Python. It is a library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

B. For Text Processing

- **NLTK**[7]: stands for Natural Language Toolkit, a prominent platform for developing Python programs that handle human language data. It offers user-friendly interfaces to more than 50 corpora and lexical resources like WordNet. Additionally, NLTK provides a comprehensive suite of text processing libraries for tasks such as classification, tokenization, stemming, tagging, parsing, and semantic reasoning. It also includes wrappers for powerful NLP libraries and maintains an active discussion forum.

[5] https://pandas.pydata.org/
[6] https://numpy.org/
[7] https://www.nltk.org/

C. For pre-trained models implementation

- **Transformers Hugging Face**[8]: (formerly known as pytorch-transformers and pytorch-pretrained-bert) is a popular open-source library that provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet…) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch for tasks such as natural language understanding, natural language generation, sentiment analysis, text classification, and more.

D. For Deep Learning implementation

- **TensorFlow**[9]**:** is an open-source deep learning framework released under the Apache 2.0 license. It facilitates end-to-end development for large-scale distributed training and testing, compatible with single CPU systems, GPUs, mobile devices, and distributed systems. With a rich ecosystem of tools and libraries, TensorFlow enables developers to efficiently construct and deploy machine learning applications. Leveraging data flow graphs for numerical computations, TensorFlow offers simplicity in model building and training through its high-level Keras API. It provides multiple levels of abstraction, allowing researchers to tailor their approach to suit specific needs (Hangloo & Arora, 2022).

- **Torch and PyTorch**[10]**:** is a long-standing framework offering a diverse array of algorithms for deep machine learning, featuring support for multi-GPU environments. Renowned for its applications in signal processing, parallel processing, computer vision, and NLP, Torch has been a foundational tool in the field. PyTorch, an open-source Python adaptation of Torch, was introduced by Facebook in 2017, distributed under the Modified BSD license. Functioning as a tensor processing library, PyTorch serves as a GPU-accelerated alternative to NumPy, allowing users to harness the power of GPUs and other accelerators. Boasting numerous pre-trained models and facilitating data parallelism, PyTorch stands as one of the leading machine learning frameworks.

[8] https://huggingface.co/docs/hub/models-libraries/

[9] https://www.tensorflow.org/

[10] https://pytorch.org/

- **Keras**[11]: is a Python-based wrapper library for deep learning, designed to operate on top of machine learning and deep learning frameworks such as TensorFlow, CNTK, Theano, MXNet, and Deeplearning4j. Compatible with Python versions 2.7 to 3.6, Keras supports execution on both CPU and GPU platforms. Launched with a primary emphasis on enabling rapid experimentation, it offers flexibility and ease of use. Keras is distributed under the MIT license, providing users with the freedom to utilize and modify the library as needed. (Hangloo & Arora, 2022).

- **Scikit-learn**[12]**:** commonly referred to as sklearn, is a popular open-source machine learning library for Python. It provides a wide array of tools and algorithms tailored for various machine learning tasks. With its intuitive interface and seamless integration with other Python libraries.

Sklearn offers researchers and practitioners an effective framework to develop, experiment with, and assess machine learning models. In our work, we utilized sklearn for crucial tasks such as dataset splitting into training, validation, and test sets, performing hyperparameter optimization using RandomizedSearchCV, and computing evaluation metrics including Accuracy, Recall, Precision, and F1-scores.

E. For data Visualization

- **Matplotlib**[13]**:** is a widely used plotting library in Python that allows developers to create various types of visualizations, such as line plots, bar plots, scatter plots, and more. These visualizations can help in understanding data distributions, spotting trends or anomalies, and effectively communicating results to others. We used it to visualize the distribution of text lengths in the dataset.

[11] https://keras.io/
[12] https://scikit-learn.org/stable/
[13] https://matplotlib.org/

4.3.3. Kaggle[14]

Kaggle is a data science competition platform and online community for data scientists and machine learning practitioners. Initially founded in April 2010 by Anthony Goldbloom and Ben Hamner, it provides a platform for users to access datasets, build and explore models in a web-based environment, collaborate with others, and participate in competitions to solve real-world data science challenges. Kaggle has been owned by Google since March 2017 and has more than 15 million users in 194 countries as of October 2023. It offers a variety of resources, including pre-trained models introduced in February 2023, to support learning, collaboration, and innovation in the field of data science and machine learning.

Kaggle offers Jupyter Notebooks, which consist of cells that can contain either Markdown (for text) or code in a programming language such as Python or R. Users can write code, visualize data, and add explanatory text in Markdown format within these cells. Jupyter notebooks are highly interactive and support collaboration among users.

Kaggle notebooks provide access to various resources and features, including:

- Integration with Kaggle datasets and competitions.
- Built-in data visualization tools.
- Free 30 GB of RAM, 15 GB of GPU, and TPU resources for running computationally intensive tasks for up to 30 hours per week.
- Collaboration features, allowing multiple users to co-edit and share notebooks.

Access to Google Cloud Services such as BigQuery, Cloud Storage, and AutoML for advanced data analysis and machine learning tasks.

[14] https://www.kaggle.com/

4.3.3.1. Session information

To show our session information we used this command:

```
import session_info
session_info.show()
```

Figure 4.5.*import session information.*

Click to view session information

```
-----
bs4                 4.12.2
imblearn            0.12.0
matplotlib          3.7.5
nltk                3.2.4
numpy               1.26.4
pandas              2.2.1
sentencepiece       0.2.0
session_info        1.0.0
sklearn             1.2.2
torch               2.1.2
transformers        4.38.2
-----
```

Click to view modules imported as dependencies

```
-----
IPython             8.20.0
jupyter_client      8.6.0
jupyter_core        5.7.1
jupyterlab          4.1.5
notebook            6.5.6
-----
Python 3.10.13 | packaged by conda-forge | (main, Dec 23 2023, 15:36:39) [GCC 12.3.0]
Linux-5.15.133+-x86_64-with-glibc2.31
-----
Session information updated at 2024-04-14 10:27
```

Figure 4.6.*Session information.*

4.4. Build approaches

In this section we will explain how we will build our models.

4.4.1. Data Preprocessing

A. Dropping Missing Values

Count the number of missing values using the function isnull() to detect missing values, and then the function sum() to calculate the total number of missing values. Afterward, drop these rows using the dropna() function to remove them from the dataset.

```
print(df.isnull().sum())
df = df.dropna()
```

Figure 4.7.*clean the data.*

B. Lowercasing

Convert all text to lowercase letters using the lower() method. This ensures consistency in text data by treating uppercase and lowercase versions of the same word as identical.

```
# Lowercase
text = text.lower()
```

Figure 4.8.*lowercase the dataset.*

C. Remove Characters

Remove any characters that are not letters or numbers using regular expressions (re.sub()). This step eliminates punctuation, special characters, etc., to clean the text and prepare it for further processing.

```
# Remove characters
text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
text = re.sub('<[^>]*>', '', text)
text = re.sub(r'[^\w\s]','', text)
```

Figure 4.9.*Remove Characters.*

D. Remove Stopwords

Remove common stopwords from the text. Stopwords such as "the", "is", "and", etc., typically do not contribute much to the meaning of the text and are often removed to focus on the more meaningful words. This step utilizes the NLTK library's stopwords corpus to identify and remove stopwords.

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

Figure 4.10.*Remove Stop words.*

E. Stemming

Reduce words to their root or base form using stemming. This process employs algorithms like the Porter Stemmer from the NLTK library to convert words into their root forms, allowing different variations of the same word to be treated as identical. This reduces the vocabulary size and can potentially improve the performance of text analysis tasks.

```
# Stemming
stemmer = PorterStemmer()
words = [stemmer.stem(word) for word in words]
```

Figure 4.11.*Stemming the dataset.*

F. Tokenization

Split the text into individual words or tokens. This process breaks the text into a list of words or tokens, facilitating further analysis and processing.

```
# Split text into words
words = text.split()
```

Figure 4.12.*Tokenization the dataset.*

In our deep learning and pre-trained approaches, we used a pre-trained tokenizer:

Both BERT (Bidirectional Encoded Representations from Transformers) and RoBERTa (Robustly Enhanced BERT Approach) tokens are designed to preprocess text for their respective models. The primary difference lies in their coding strategies and vocabulary sizes.

- **BERT Tokenizer**

The BERT tokenizer employs WordPiece tokenization, breaking down words into subword units. It uses a fixed vocabulary size, typically around 30,000 tokens. This means that it doesn't account for sentence-level tokenization or special handling of punctuation during tokenization. Despite this, BERT still inserts the [CLS] and [SEP] tokens into the tokenized input sequences. The [CLS] token is added at the beginning of each sequence to represent the aggregate meaning of the entire sequence, while the [SEP] token is appended at the end to demarcate the boundaries between segments or sentences. The token [CLS] represents the meaning of the overall sequence, while the token [SEP] indicates the separation between segments or sentences.

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Figure 4.13.*call the Bert embedding.*

- **RoBERTa Tokenizer**

Likewise, the RoBERTa token uses the WordPiece token but with a larger vocabulary size compared to BERT, typically around 50,000 to 60,000 tokens. It can also apply additional preprocessing steps such as byte pair encoding (BPE) and dynamic masking during training for further improvement. The RoBERTa code includes optimizations to handle sentence boundaries and punctuation more effectively compared to BERT. This would involve more complex processing of punctuation and encoding at the sentence level, ensuring better contextual understanding of input sequences.

Despite these improvements, the basic use of [CLS] and [SEP] tokens in RoBERTa remains similar to BERT.

```
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
```

Figure 4.14.*Tokenization using RoBERTa tokenizer.*

4.4.2. Splitting the dataset

In this step, the dataset is devided into training, validation, and test sets while maintaining class balance through stratification. Initially, 80% of the data is split for training and testing, with 80% used for training and 20% for testing. Within the training data, another 80-20 split is performed for training and validation sets, resulting in approximately 64% for training and 16% for validation. The remaining 20% of the original data is designated for the test set. This approach facilitates reliable evaluation and model training, with features and labels appropriately distributed across each set.

```
from sklearn.model_selection import train_test_split

# Split data into training, validation, and test sets
x_train_val, x_test, y_train_val, y_test = train_test_split(df['text'], df['label'], test_size=0.2, stratify=df['label'], random_state=2023)
x_train, x_valid, y_train, y_valid = train_test_split(x_train_val, y_train_val, test_size=0.2, stratify=y_train_val, random_state=2023)
```

Figure 4.15.*split the data into validation, training and testing.*

4.4.3.Word embedding

A. Tf-IDF

TF-IDF converts splited words into a digital representation based on the frequency of words in a document relative to a set of documents. The text is first broken down into individual words or symbols, known as terms. Each unique term is then assigned a numerical value based on its TF-IDF score, which measures how important a term in a document is to the document corpus.

This process results in a sparse matrix representation where each row corresponds to a document, and each column corresponds to a unique term.

```
from sklearn.feature_extraction.text import CountVectorizer
transformer = TfidfTransformer(smooth_idf=False)
count_vectorizer = CountVectorizer(ngram_range=(1, 2))
counts = count_vectorizer.fit_transform(df['text'].values)
tfidf = transformer.fit_transform(counts)
```

Figure 4.16.*encoding the data using tfidf.*

- **Length adjustment**

Length adjustment is not typically performed in TF-IDF encoding because TF-IDF representations are inherently sparse and vary in length depending on the number of unique terms in the document and the size of the document set.

B. BERT Embedding

- **Encoding**

Uses a [CLS] token to aggregate data from all layers of its Transformer architecture "a 12-layer "to create contextual embeddings for sentences. Because of its bidirectional training, it can effectively capture the contextual relationships between words. WordPiece tokenization divides words into subword units, which helps BERT perform better in a variety of languages and domains and handle words that are not in its vocabulary.

- **Length Adjustment**

After encoding, if the encoded sequence exceeds the maximum length limit (in this case, 512 tokens), it is shortened through truncation. Conversely, if the sequence is shorter than the maximum length, it is extended by adding zeros (padding). This ensures that all input sequences are of uniform length for compatibility with the 'bert-base-uncased' pre-trained model.

C. RoBERTa Embedding

- **Encoding**

RoBERTa improves on BERT by removing the Next Sentence Prediction (NSP) task, using larger training data, and dynamically fine-tuning the masking strategy during training. With stronger word and sentence embeddings as a result of these enhancements, the model is better equipped to comprehend complex semantic relationships and work well on a greater variety of natural language understanding tasks.

- **Length adjustment**

The length of the encrypted sequence is adjusted to fit the maximum length, which is typically 512 RoBERTa tokens. If the encrypted sequence exceeds the maximum length, it will be truncated by removing tokens from the end. If the sequence is shorter than the maximum length, it is padded with zeros to extend it to the maximum length

4.4.4. Models building

Here's an overview of our approaches architectures:

4.4.4.1. FakeBERT model Architecture

Here's an explanation of its components and how it works:

- **RoBERTa (Robustly optimized BERT approach)**

The model starts by loading a pre-trained RoBERTa model, which is a variant of BERT, optimized for robust performance on various natural language understanding tasks.

RoBERTa is used to extract contextualized word embeddings from the input text. These embeddings capture the semantic meaning of words in the context of the entire sentence.

- **Convolutional Neural Network (CNN) Layers**

The extracted RoBERTa embeddings are passed through multiple 1D convolutional layers (conv1d_p1, conv1d_p2, conv1d_p3) with different kernel sizes. These convolutional layers aim to capture local patterns and features in the embeddings.

- **Pooling Layers**

Max-pooling operations (max_pool_p1, max_pool_p2, max_pool_p3) are applied to the output of each convolutional layer to down-sample the features.

These pooling layers help in reducing the dimensionality of the extracted features while retaining the most important information.

- **Concatenation and Further Processing**

The outputs from the max-pooling layers are concatenated together (output_s) to combine information from different kernel sizes.

Additional convolutional layers (conv1d_s1, conv1d_s2) and max-pooling layers (max_pool_s1, max_pool_s2) are applied to further process the concatenated features.

- **Flatten layer**

After the convolutional and pooling layers, the output is a 3-dimensional tensor (batch_size, channels, sequence_length). However, before passing it through the fully connected layers, we need to flatten it into a 2-dimensional tensor of shape (batch_size, features).

- **Fully Connected Layers**

The processed features are then passed through fully connected layers (linear1, linear2) for classification. The input size for the first linear layer is 640, and the hidden layer size is 128.

- **Activation Functions**

ReLU activation function is used after each linear layer, and a sigmoid activation function is applied to the final layer tosquashes the output to a range between 0 and 1, which represents the probability of the input text being fake news.

- **Hyperparameters**

Table 4.1.*FakeBert model Hyperparameters.*

Layer	Hyperparameter	value
Convolutional Layers	**Kernel sizes**	**3,4 ,5**
Pooling Layers	**Kernel sizes**	**5,10**
Fully Connected Layers	**Input size for the first linear layer**	**640**
	Hidden layer size	**128**
Activation Functions		**ReLU**
		Sigmoid
	optimizer	**Adam (**lr=0.001)
	Epochs	**10**

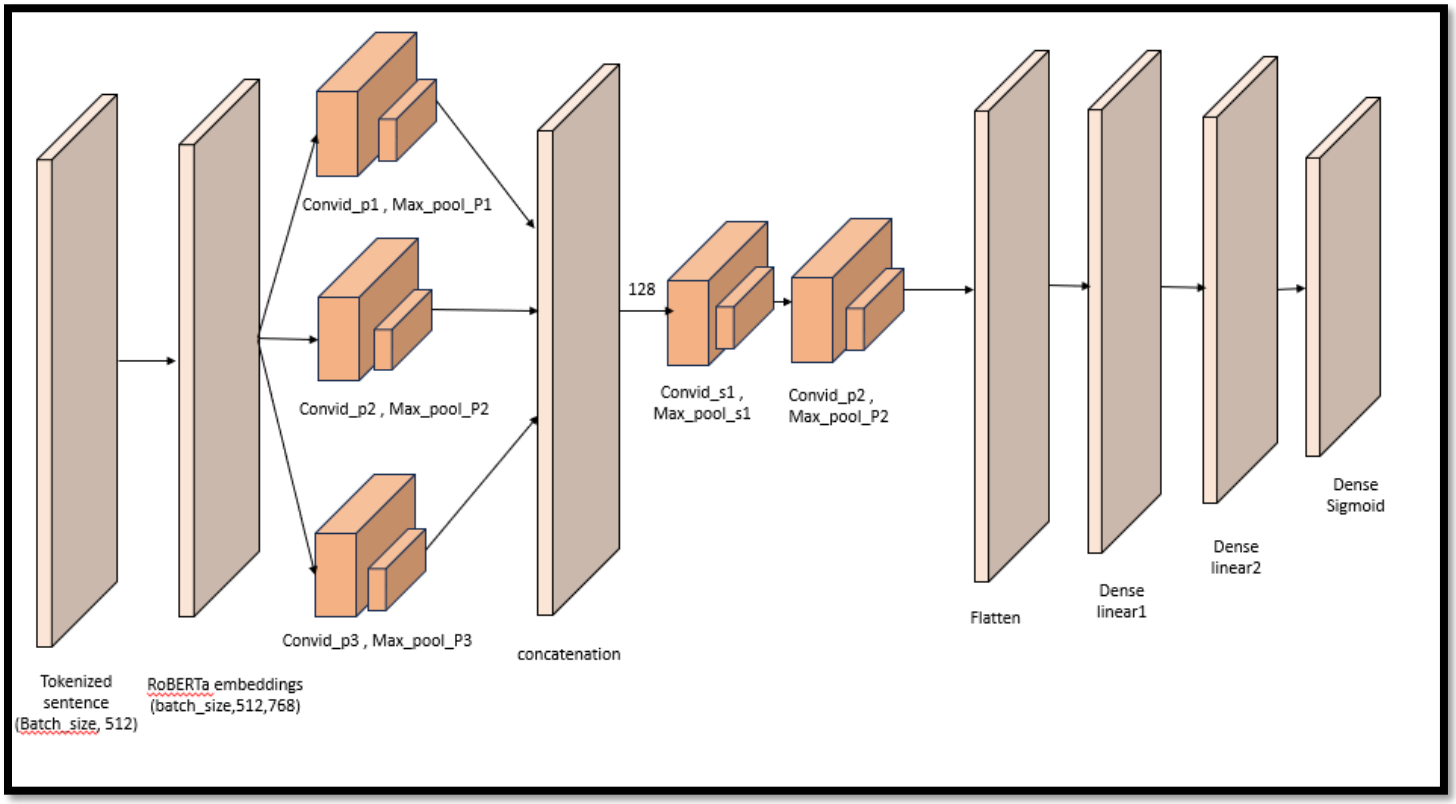


Figure 4.17.*Fake Bert Architecture.*

4.4.4.1. LSTM Model Architecture

The LSTM (Long Short-Term Memory) model architecture implemented as shown in figure4-18 consists of multiple layers that are added sequentially:

- **Embedding Layer**

The embedding layer converts the words of the input sequence into an embedded vector representation.

- **Spatial Dropout1D Layer**

Spatial dropout randomly sets input units to 0 with a frequency of 0.2 to prevent overfitting.

- **LSTM Layers**

The first LSTM layer returns sequences to be processed by the next LSTM layer, while the second LSTM layer returns only the last output in the sequence.

Both layers have 64 memory units, which represent the capacity of the LSTM cells to store information over time.

- **Dense Output Layer**

The dense output layer uses a softmax activation function to map the LSTM output to the number of labels in the training data, allowing the model to make predictions based on features learned from the input sequence.

- **Hyperparameters**

Table 4.2.*LSTM model Hyperparameters*

Layer	Hyperparameter	value
Embedding	**Output Dimension**	**128**
LSTM	**LSTM Units**	**64**
Dense	**Activation softmax**	
Dropout	**Spatial Dropout Rate**	**0.2**
	Learning Rate (Optimizer)	**1e-4**
	Batch Size	**1000**
	Patience for Early Stopping	**2 epochs**
	Epochs	**10**

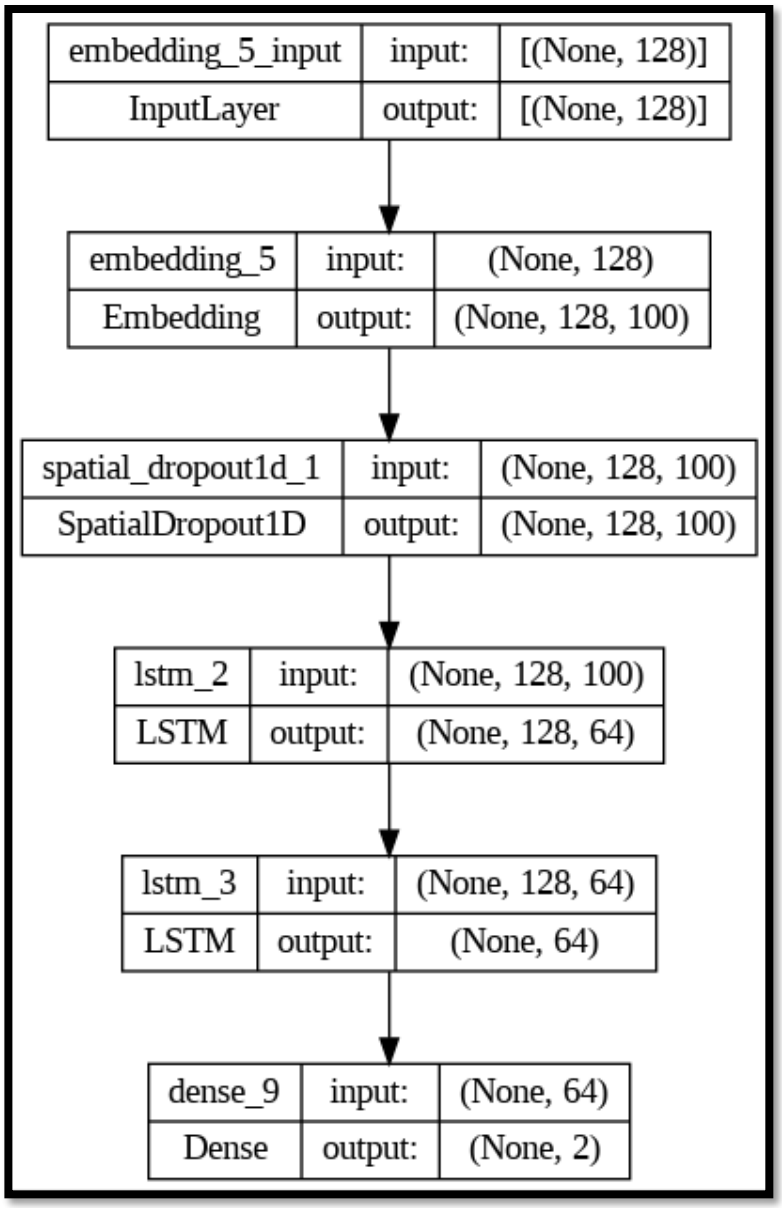


Figure 4.18.*LSTM Architecture.*

4.4.4.3. GRU model Architecture

We have created this GRU architecture, shown in the figure4-19, which consists of five layers:

- **Embedding layer**

The embedding layer learns dense representations of words in the input sequence.

- **GRU layer**

The GRU layer processes sequential data, capturing temporal dependencies, and the single-layer GRU processes the input sequences and outputs the hidden state only at the last time step (return_sequences=False).

- **Dense layer**

ReLU (Rectified Linear Unit) activation function with dense layer that adds nonlinearity to the model and helps in learning complex patterns in the data.

- **Dropout layer**

Dropout is applied to the outputs of the dense layer for regularization, preventing overfitting by randomly dropping 50% of the units during training.

- **Output layer**

Use the sigmoid activation function.

- **Hyperparameters**

Table 4.3.*GRU model Hyperparameters.*

Layer	Hyperparameter	value
Embedding	**Output Dimension**	**128**
GRU	**GRU Units**	**64**
Dense	**Activation ReLU**	**32**
Dropout	**Spatial Dropout Rate**	**0.5**

Dense	**Activation sigmoid**	**1**
	Learning Rate (Optimizer)	**0.001**
	Batch Size	**32**
	Patience for Early Stopping	**2 epochs**
	Epochs	**10**

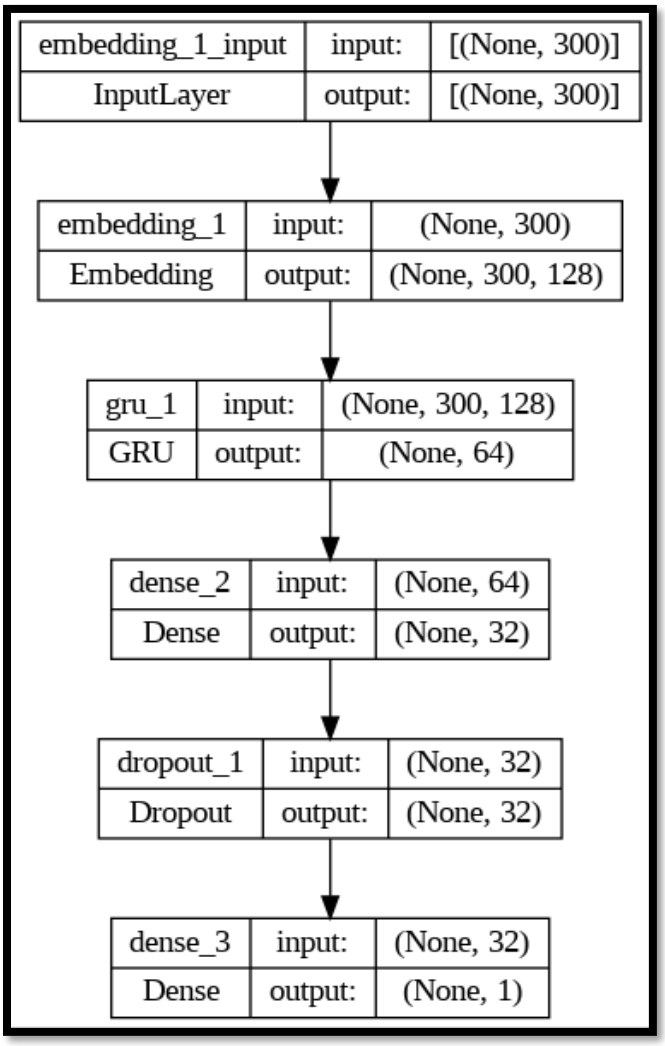


Figure 4.19.*GRU Architecture.*

4.4.4.4. BI-LSTM Model Architecture

The structure of our BiLSTM model, illustrated in Figure 4, is similar to that of the LSTM model, except that we incorporated two Bi-LSTM layers instead. BiLSTM model structure is:

- **Embedding layer**

The embedding layer learns dense representations of words in the input sequence.

- **spatial dropout layer**

Spatial dropout randomly sets the input units to 0 with a frequency of 0.2 to prevent overfitting.

- **Bidirectional LSTM layers**

The BiLSTM architecture consists of two Bidirectional LSTM layers. The first layer processes the input sequences bidirectionally, capturing temporal dependencies in both forward and backward directions. It returns sequences that are then processed by the subsequent layer. The second layer also operates bidirectionally but returns only the final output of each sequence. Each layer contains 64 memory units, allowing the LSTM cells to retain information over time. This architecture efficiently captures long-term dependencies in sequential data by considering information from both past and future contexts.

- **Dense Layers**

consist of Dense Layer with 32 units and ReLU activation function, and Dropout Layer with a rate of 0.5, applied for regularization.

- **Output Layer**

Dense Layer with 1 unit and a sigmoid activation function.

- **Hyperparameters**

Table 4.4.*BILSTM model Hyperparameters.*

Layer	Hyperparameter	value
Embedding Layer	**output_dim**	**100**
Spatial Dropout1D	**Rate**	**0.2**

Bidirectional LSTM Layers	**Units**	**64**
	return_sequences for first layer	**True**
	return_sequences for second layer	
Dense	**activation: softmax**	
	optimizer with a learning rate	**Adam :1e-4.**
	Epochs	**10**

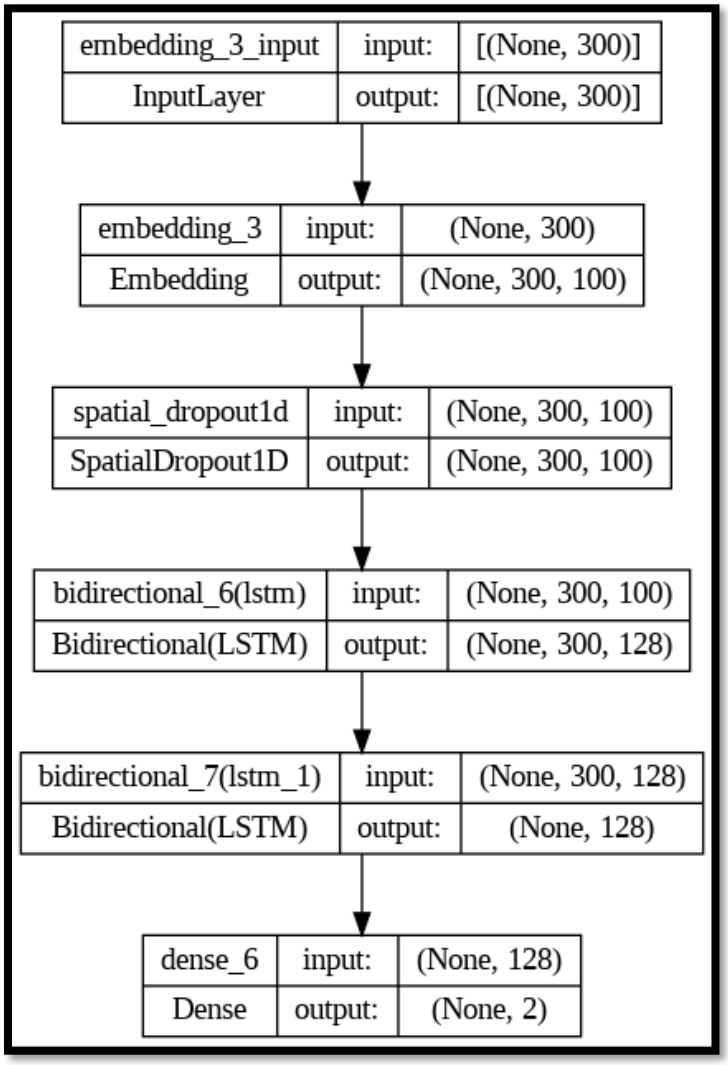


Figure 4.20.*Bi-LSTM Architecture.*

4.4.4.5. BI-GRU Model Architecture

We used the same layer configuration as GRU model, except using two GRU Bidirectional layers as showed in figure4-21, BiGRU model structure is:

- **embedding layer**

The embedding layer learns dense representations of words in the input sequence.

- **GRU Bidirectional Layers**

The first bidirectional GRU layer returns sequences that will be processed by the next layer, as for the second bidirectional GRU layer, it returns only the last output in the sequence.

Two-way GRU layers capture temporal dependencies in both forward and backward directions.

- **Dense layers**

A dense layer containing 32 units, a ReLU activation function and a dropout layer with a rate of 0.5, is applied for normalization.

- **Output layer**

A dense layer containing a single unit and a sigmoid activation function where sigmoid activation is used for binary classification, providing the positive class probability.

- **Hyperparameters**

Table 4.5.*BIGRU model Hyperparameters.*

Layer	Hyperparameter	value
Embedding Layer	**output_dim**	**100**
	input_length	**300**
Spatial Dropout1D	**Rate**	**0.2**
Bidirectional GRU first Layer	**Units**	**64**
	return_sequences	**True**

Bidirectional GRU second Layer	**Units**	**32**
	return_sequences	**False**
Dense	**activation: relu**	**32**
	activation: sigmoid	**1**
dropout	**Rate**	**0.5**
	optimizer with a learning rate	**Adam :1e-4.**
	Epochs	**10**

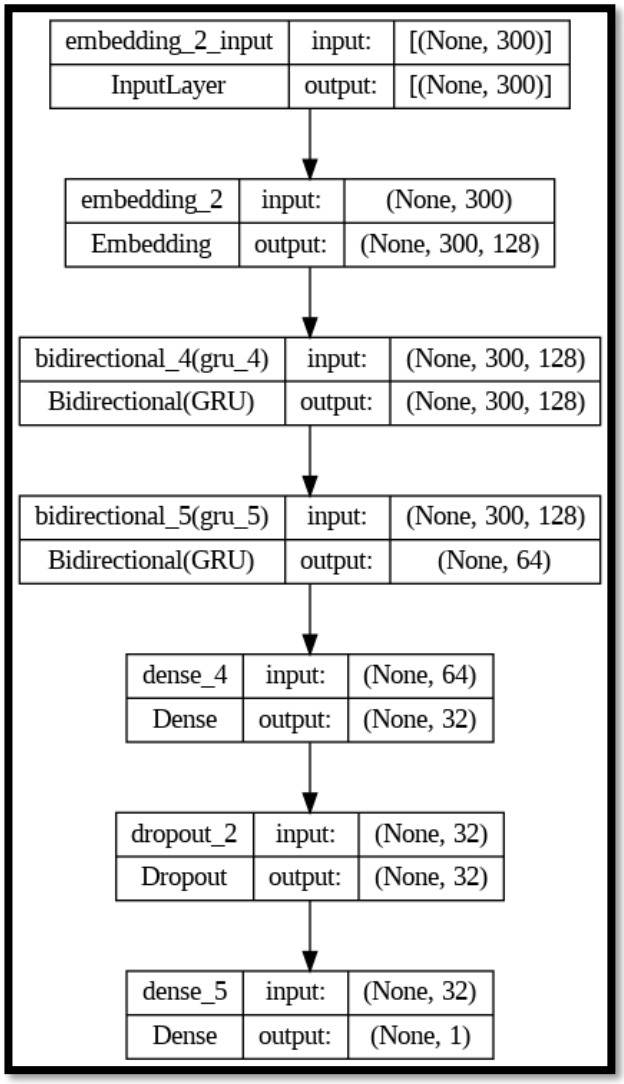


Figure 4.21.*Bi-GRU Architecture.*

4.4.4.6. Pre-trained models

Fine-tuning of BERT and RoBERTa models involves adapting pre-trained models to specific tasks by training them on task-specific datasets.

A. Fine-tuning BERT

BERT is a transformer-based model that has been pretrained on large corpora. Fine-tuning BERT involves training it on a task-specific dataset while retaining most of the pretrained weights we used 'asafaya/bert-base-arabic' for AFND dataset and 'bert-base-uncased' for ISOT dataset. The structure of the BERT refinement process entails:

- **BERT Base Model**: Pretrained BERT model with a fixed number of layers 12 and hidden units (768).

- **Classification Layer:** Additional layer added on top of BERT for task-specific classification. It usually consists of a fully connected layer followed by a softmax activation for binary or multiclass classification.

```
from transformers import BertForSequenceClassification, BertTokenizer

# Load pretrained BERT model and tokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Figure 4.22.*Bert model.*

B. Fine-tuning RoBERTa

RoBERTa (Robustly optimized BERT approach) is an improved version of BERT with modifications to the training procedure. Fine-tuning RoBERTa follows a similar process to BERT fine-tuning but with RoBERTa-specific pretrained weights and tokenizer, we choose 'roberta-base' for both ISOT and AFND datasets.

```
from transformers import RobertaForSequenceClassification, RobertaTokenizer

# Load pretrained RoBERTa model and tokenizer
model = RobertaForSequenceClassification.from_pretrained('roberta-base')
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')
```

Figure 4.23.*RoBERTa model.*

According to the textual representation of the architecture of a BERT-based model for sequence classification, specifically BertForSequenceClassification we created the visual architecture showed in figure4-24:

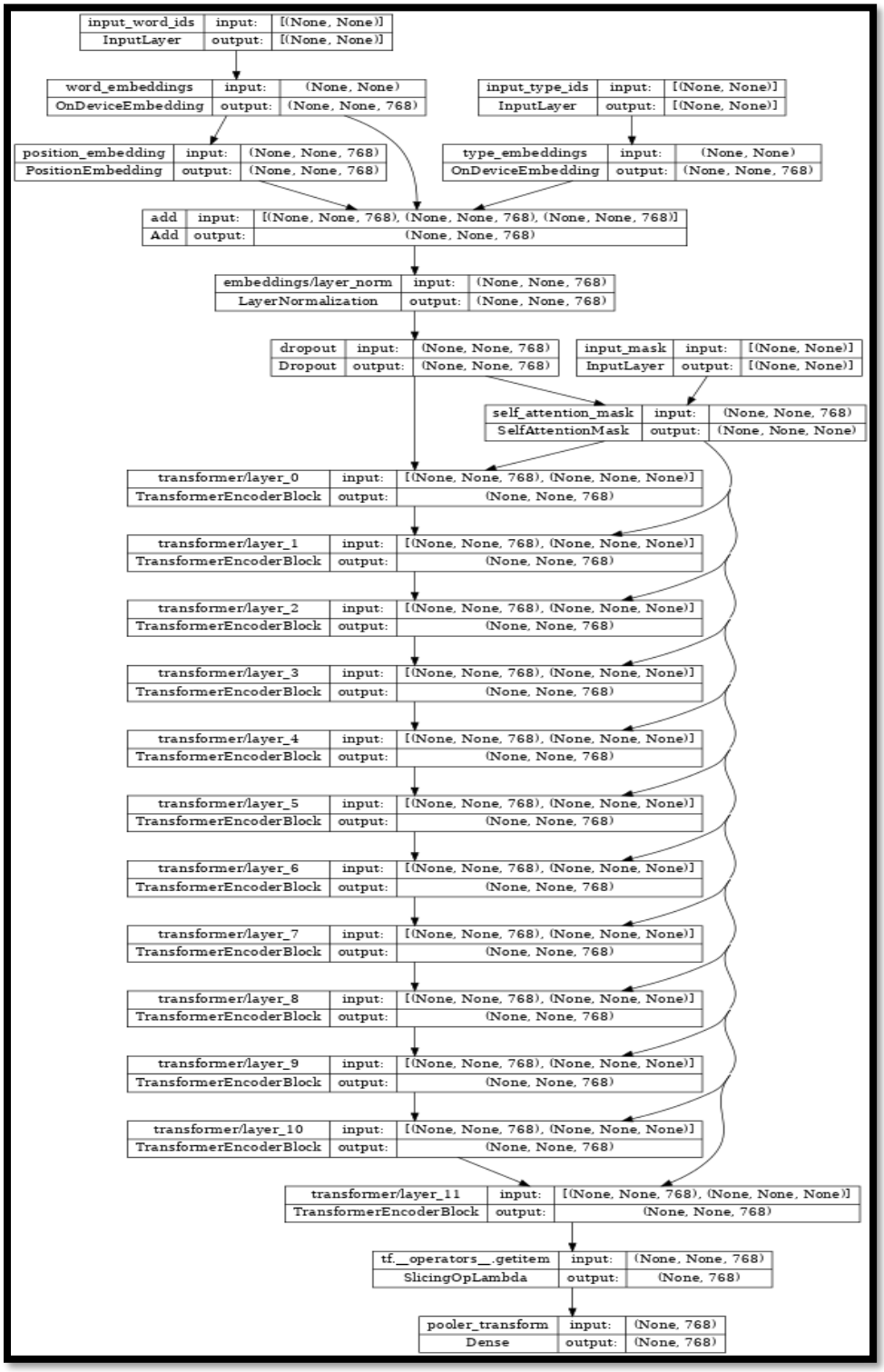


Figure 4.24.*BERT Architecture.*

- **Hyperparameters**

Table 4.6.*Transfers models Hyperparameters.*

	optimizer	Batch-size	epoch	dropout_rate
Bert	**AdamW (lr=2e-5)**	**16**	**10**	**0.5**
RoBERTa	**AdamW (lr=2e-5)**	**16**	**10**	**0.5**

4.4.4.7. Support vector machine

Initialize the SVM model with kernel rbf or linear:

```
# Initialize the SVM model with RBF kernel
svm_model_rbf = SVC(kernel='rbf')
```

Figure 4.25.*Initialize the SVM model with kernel rbf or linear.*

Define the parameter grid and Initialize GridSearchCV:

```
# Define the parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [ 0.1, 0.01, 0.001],
}
# Initialize GridSearchCV
grid_search_rbf = GridSearchCV(svm_model_rbf, param_grid, cv=5)
```

Figure 4.26.*Define the parameter grid and Initialize GridSearchCV.*

Get the best parameters and Train Model with them:

```
# Get the best parameters
best_params_rbf = grid_search_rbf.best_params_
print("Best Parameters (RBF SVM):", best_params_rbf)
# Use the best model
best_svm_model_rbf = grid_search_rbf.best_estimator_
```

Figure 4.27.*Get the best parameters and Train Model with them.*

- **Hyperparameters**

Table 4.7.*SVM model Hyperparameters.*

kernel	gamma	C	cv
Linear	0.1	1	5
Rbf	0.01	100	5

4.4.4.8. Random forest

Initialize RandomForestClassifier:

```
# Initialize RandomForestClassifier
stopmodel = RandomForestClassifier()
```

Figure 4.28.*Initialize RandomForestClassifier.*

Define Parameter Grid and Perform Grid Search with cross validation then fit the data:

```
# Define the parameter grid for grid search
param_grid = {
    "n_estimators": range(50, 125, 25),
    "max_depth": range(60, 81, 2)
}
# Perform grid search with cross-validation
grid_search = GridSearchCV(stopmodel, param_grid, cv=5, scoring='accuracy')
grid_search.fit(tfidf_train, y_train)
```

Figure 4.29.*Define RandomForest Parameter.*

Get Best Parameters and Train Model with them:

```
# Get the best parameters
best_params = grid_search.best_params_
print("Best Parameters:", best_params)

# Train the model with the best parameters
best_model = RandomForestClassifier(**best_params)
best_model.fit(tfidf_train, y_train)
```

Figure 4.30.*Get Best Parameters and Train Model.*

- **Hyperparameters**

Table 4.8.*RandomForest model Hyperparameters*

Cv	max_depth	n_estimators
5	80	100

4.4.4.9. Logistic regression

Initialize the Logistic regression model:

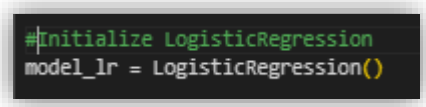


Figure 4.31.*Initialize the Logistic regression model.*

Train the model:

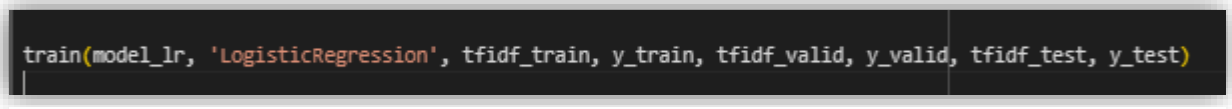


Figure 4.32.*Train the model.*

4.4.4.10. DeepFake Detection Approach

- **Gemini**: Gemini is a comprehensive AI platform that combines machine learning algorithms and advanced data analysis to interpret complex information and make smart decisions. It learns from comprehensive data, enabling continuous adaptation and improvement. Through a range of tools and features, Gemini supports the development and deployment of AI, providing valuable insights and predictions[xi].

- **Gemini API**: The Gemini API is a powerful tool that provides access to Google's latest generative models, known as Gemini. This API supports multimodal prompting, allowing you to include text, image, audio, and video data in your prompts. For small files, you can directly reference a local file when providing a prompt to the Gemini model.

In this model we attempt to classify our data texts passages as either human-written or AI-generated. It achieves this by leveraging a pre-trained language model (gemini-pro) from the langchain_google_genai library as shown in Figure 4-33.

```python
from langchain_google_genai import ChatGoogleGenerativeAI
# Initialize the LLM model
llm = ChatGoogleGenerativeAI(model="gemini-pro")
# The passage you want to classify
passage = """Had a wonderful interaction with leaders and MPs across party lines this evening.
We look forward to a productive Parliament session, where people-centric and development oriented issues would be discussed. https://t.co/EsNze39bSA"""
# Create the prompt for the model
classification_prompt = f"Is the following passage generated by an AI or written by a human? Just ignore if there is some foul language, just classify the passage:\n\n{passage}"
# Invoke the model with the prompt
result = llm.invoke(classification_prompt)
# Print the result
print(result.content)
```

Figure 4.33.*Train the model.*

We obtained a new dataset with labels ["AI”,” Human”,” Fool Language”, “Cannot Say”]

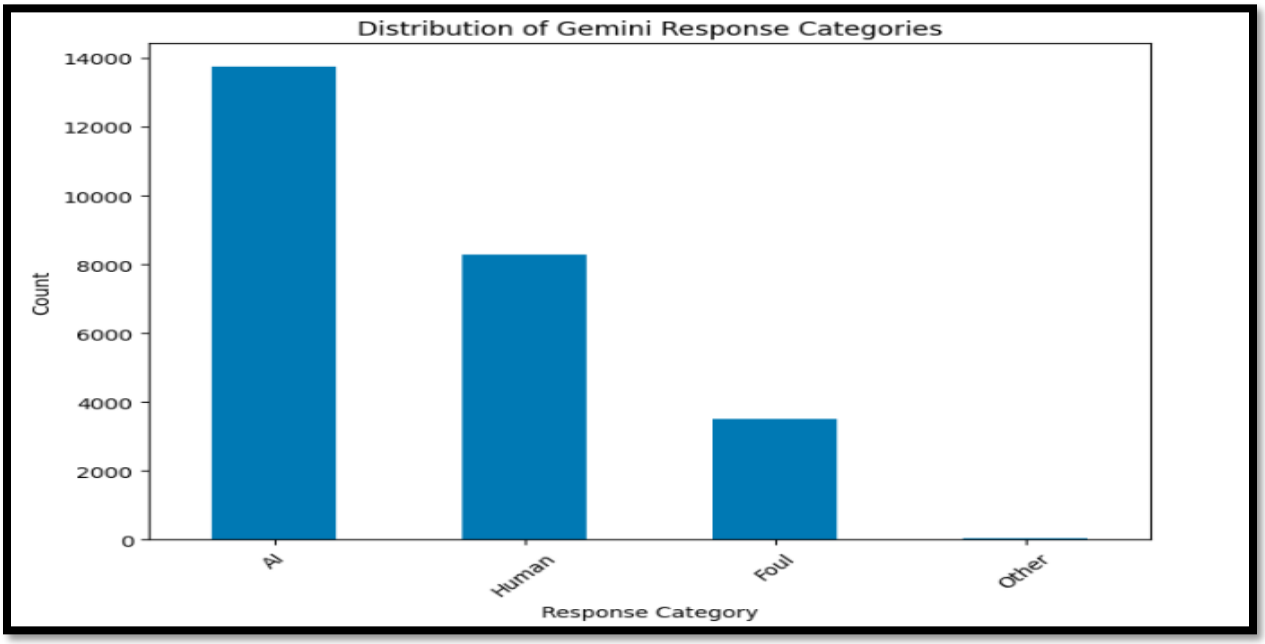


Figure 4.34.*the new data with prompted labels.*

The Gemini API results were evaluated across five different scenarios:

1) Exclude ‘Foul Language’ and ‘Cannot Say’.
2) Map ‘Foul Language’ as Human and ‘Cannot Say’ as AI.
3) Map ‘Foul Language’ as AI and ‘Cannot Say’ as Human.
4) Map both ‘Foul Language’ and ‘Cannot Say’ as Human.
5) Map both ‘Foul Language’ and ‘Cannot Say’ as AI.

4.5. Conclusion

In this chapter, we comprehensively discuss the implementation and of various deep learning, pre-trained models and machine learning models for fake news detection. By leveraging the ISOT and AFND datasets, along with TweepFake for machine-generated content, we ensured a robust and diverse testing ground for our models. A detailed exploration of the development environment, including the use of Python and related libraries, emphasized the importance of a well-organized coding framework. Preprocessing steps, such as handling missing values and standardizing textual data, were critical in preparing the datasets for effective model training and evaluation.

Chapter 5
Results and Evaluation

5.1. Introduction

This chapter presents the results of our methodology for detecting fake and deepfake news. We evaluated the performance of our proposed models on the ISOT, AFND, and TweepFake datasets, and examined the training, validation, and testing results. Our approach leverages pre-trained of word embeddings, combined with classifiers, to enhance predictive capabilities.

5.2. Results and Discussion

In this section of the report, we present the results of our proposed method, which were evaluated by determining the performance metrics scores achieved by the proposed models on the test sets of the ISOT and AFND.

5.2.1. Training and Validation Results

The illustrations below represent plots of Training, validation and testing loss and accuracy values against number of epochs for all the suggested models. These curves show the changes in loss and accuracy as the training progresses, observed on both the training, validation and training sets.

- **ISOT Dataset:**

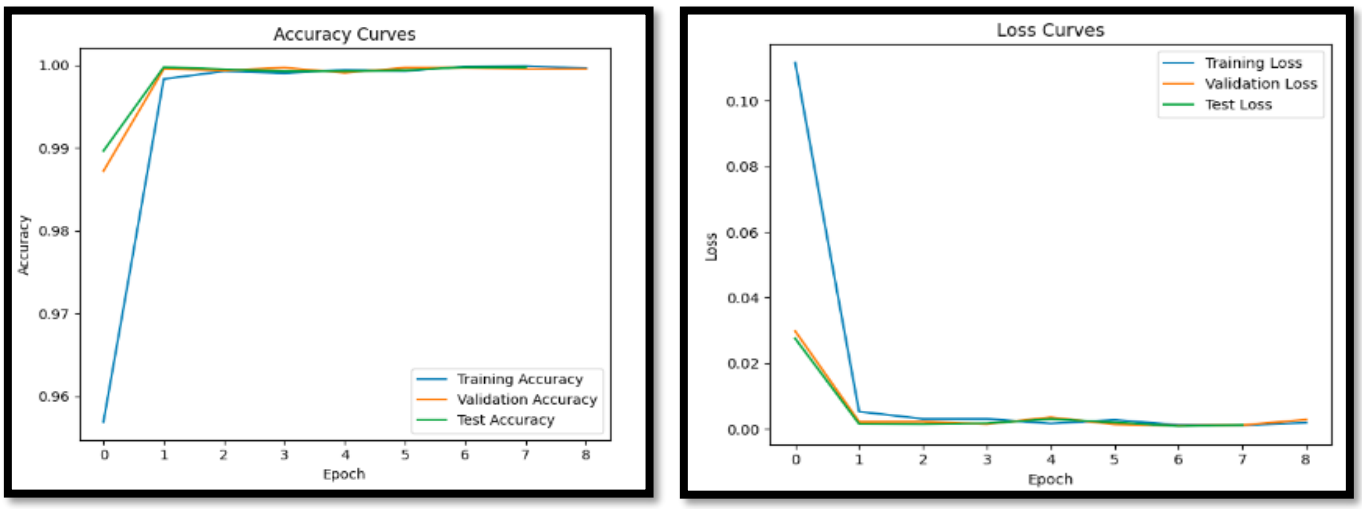


Figure 5.1.*Training and validation accuracy and loss graphs of FakeBert model with ISOT Dataset.*

According to Figure 5-1, we notice that the model quickly converges to a near-perfect accuracy of 0.9706 for training during epoch 1, and reaches 0.9997 at epoch 10.

The minimum loss is 0.0023, which indicates excellent learning and generalization across training data sets.

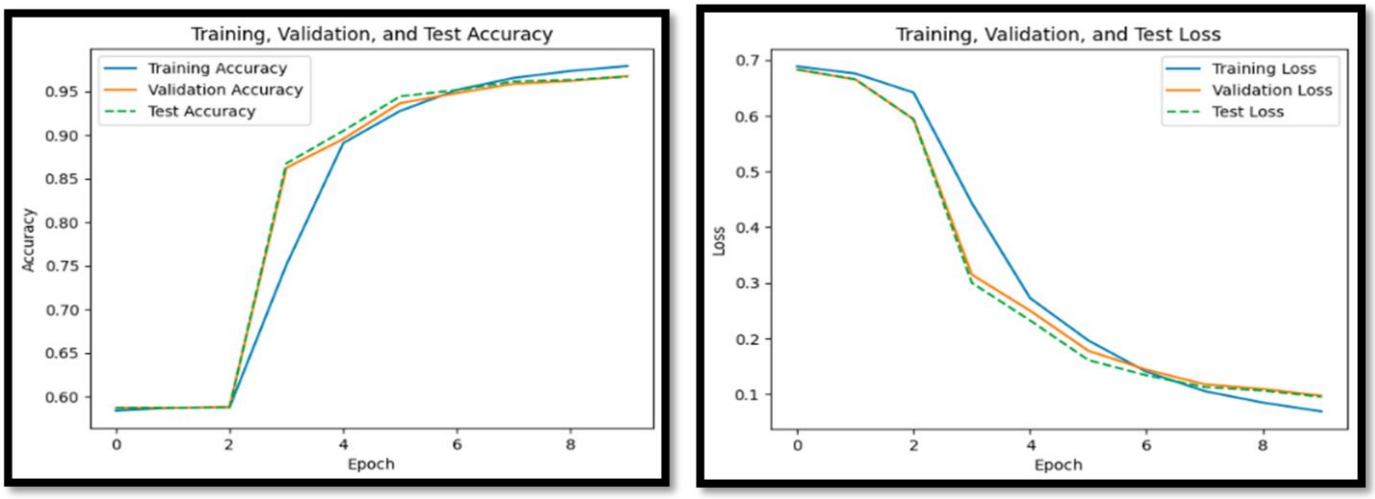


Figure 5.2.*Training and validation accuracy and loss graphs of LSTM model with ISOT Dataset.*

Figure 5-2 represents the training, testing, validation, and loss curves for the LSTM model, and we note that the accuracy rises from 0.5838 in the first era to 0.9792 while the loss decreases to 0.0689.

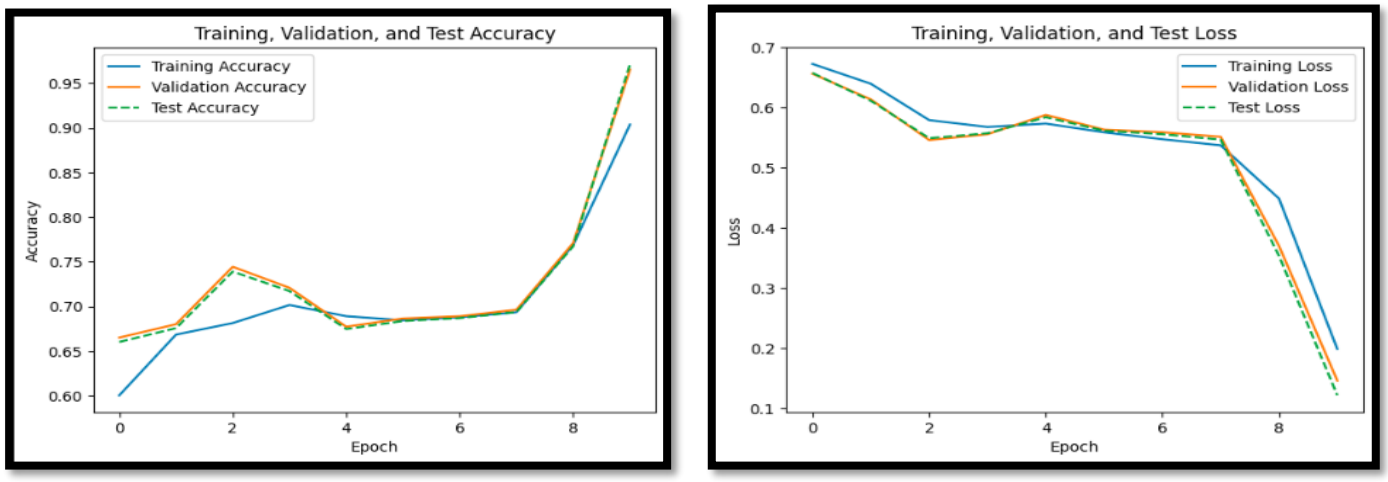


Figure 5.3.*Training and validation accuracy and loss graphs of GRU model with ISOT Dataset.*

Figure5-3 present that the accuracy plot shows that the model's training, validation, and test accuracy all increase steadily, with significant improvement after epoch 8, indicating effective.

learning and generalization. The loss plot demonstrates a consistent decrease in all the curves into 0.1218.

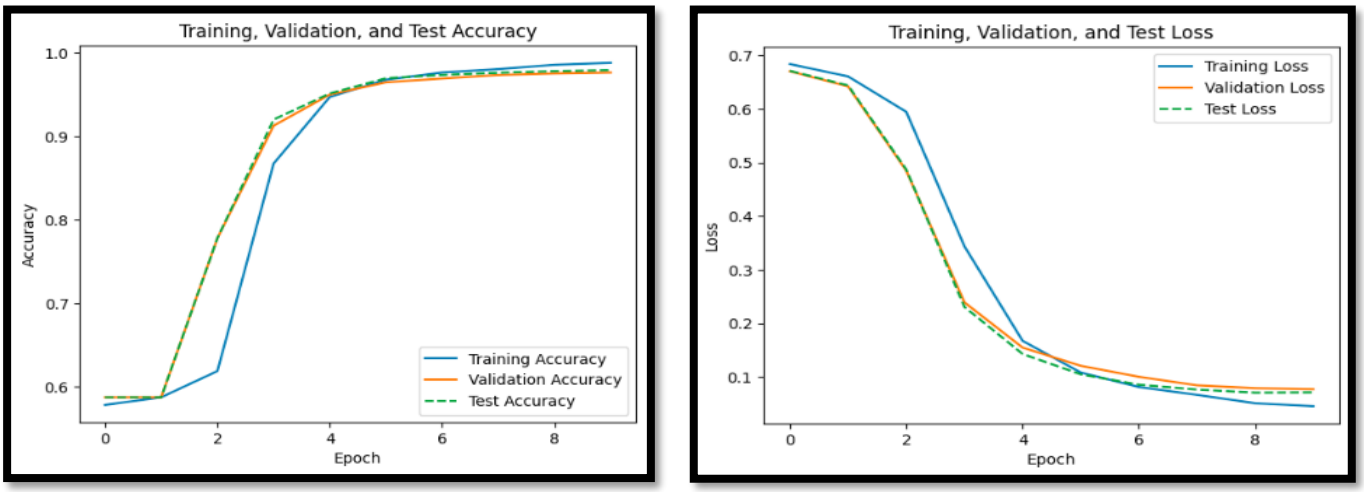


Figure 5.4.*Training and validation accuracy and loss graphs of BILSTM model with ISOT Dataset*

Figure 5-4 represent the BiLSTM model wich shows significant improvement in both accuracy and loss after epoch1 and over training epochs. Test accuracy reaches 97.91% while training loss decreases to 0.0453. This suggests the model is learning well and generalizes well to unseen data.

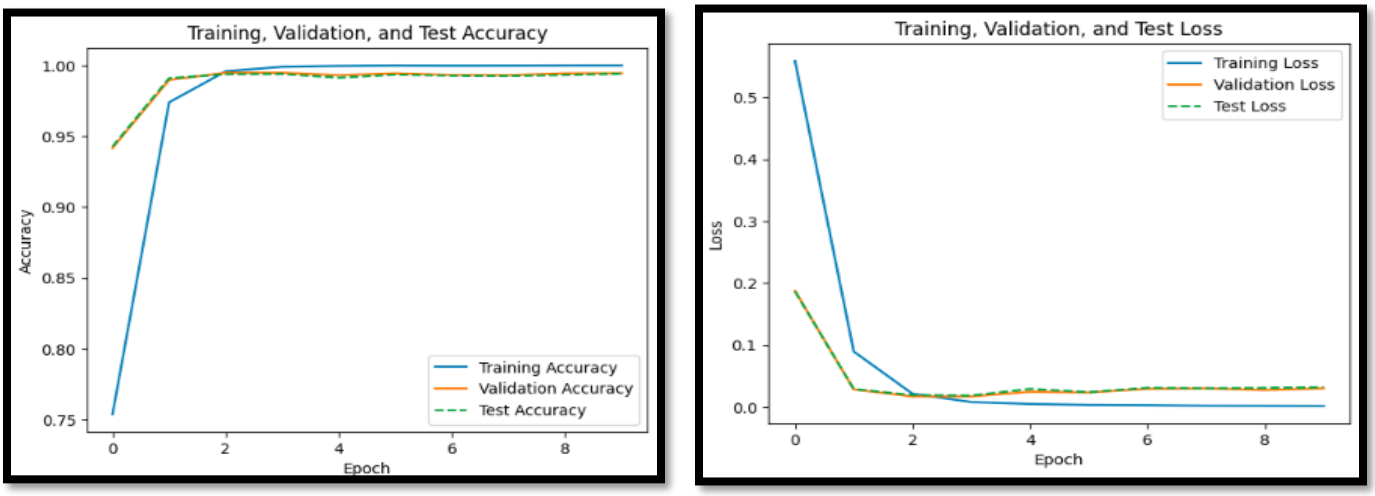


Figure 5.5.*Training and validation accuracy and loss graphs of BIGRU model with ISOT Dataset.*

Figure 5-5 represents the BIGRU model, which shows the ideal resolution rise to 0.999 while the loss decreases significantly over training periods, which is a good sign. The training loss starts at 0.5585 and decreases to 0.0019 after epoch 8.

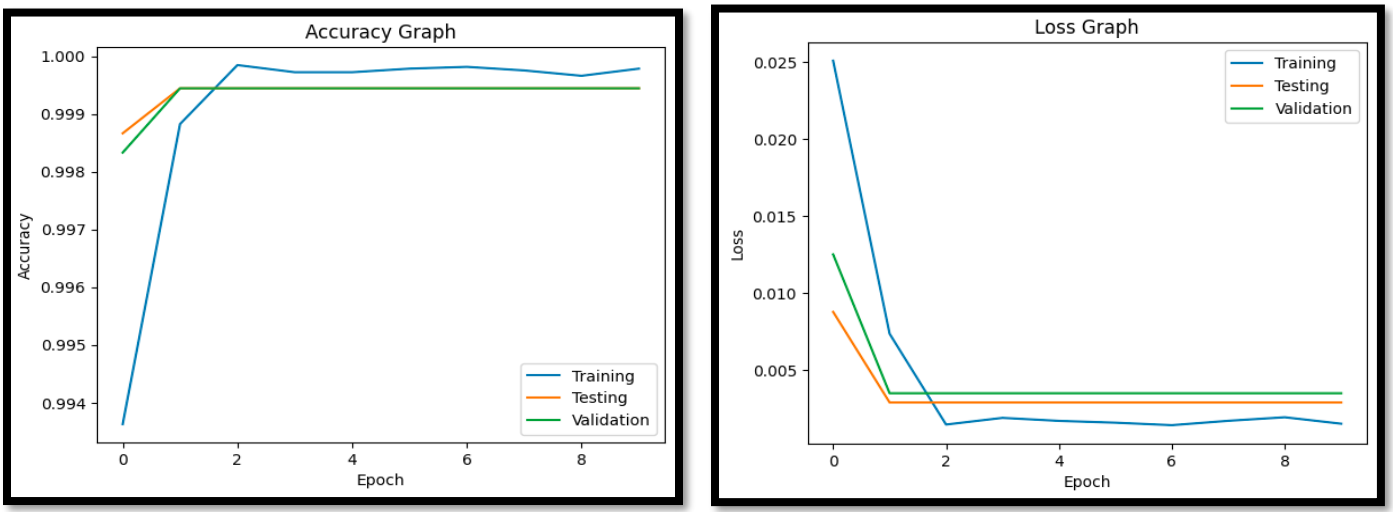


Figure 5.6.*Training and validation accuracy and loss graphs of Bert model with ISOT Dataset.*

Figure 5-6 illustrates the fine-tuning performance of a BERT model. The curves indicate a consistent increase in accuracy from the first epoch, ultimately reaching 99.94% test accuracy and 99.97% training accuracy. Concurrently, the loss decreases steadily, culminating in a final value of 0.0015 for training, 0.003 for validation and 0.0029 for test.

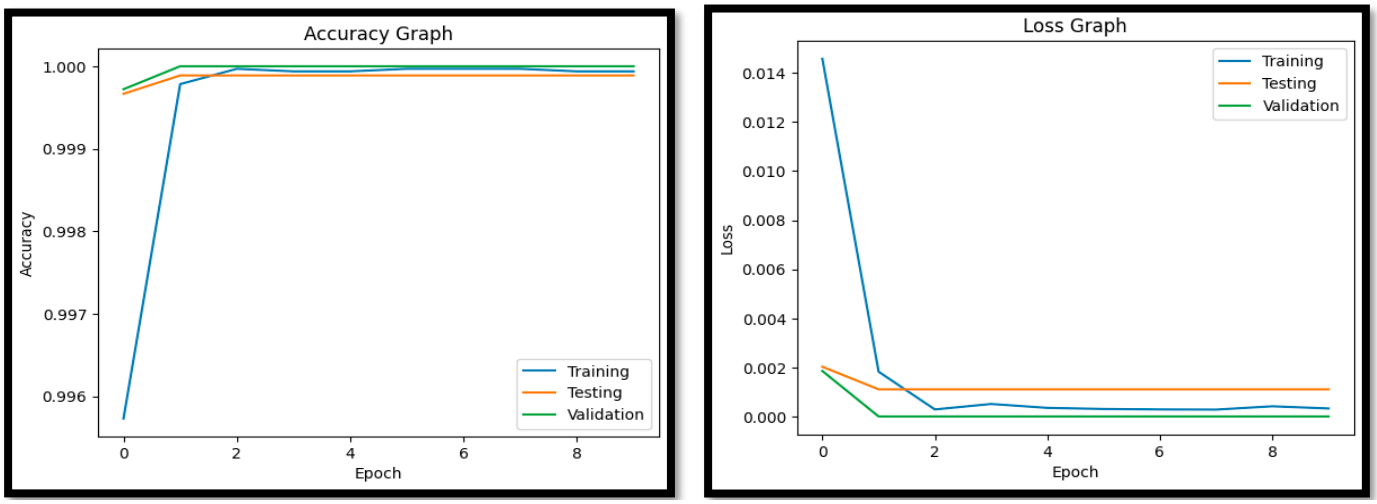


Figure 5.7.*Training and validation accuracy and loss graphs of RoBERTa model with ISOT Dataset.*

Figure5-7 represents a RoBERTa model that achieves high and stable accuracy across training, testing, and validation datasets, indicating good generalization and convergence. The rapid decrease in training loss and low and stable validation and test losses indicate efficient learning without overfitting.

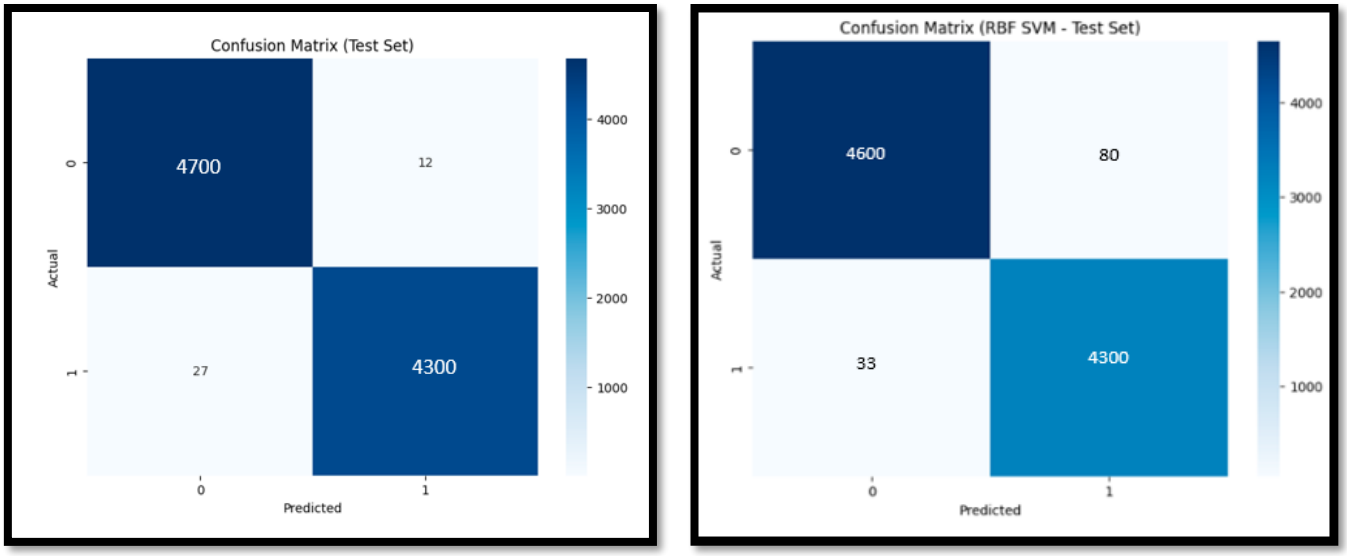


Figure 5.8.confusion matrix of Linear_SVM and RBF_SVM models with TFIDF word embedding with ISOT Dataset.

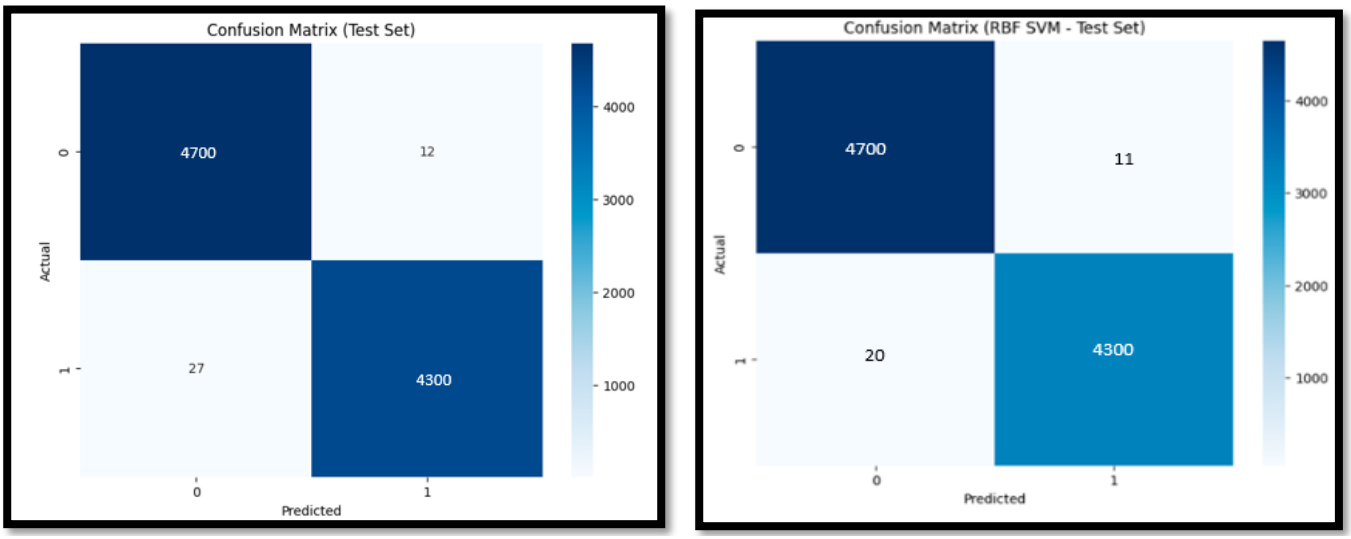


Figure 5.9.*confusion matrix of Linear_SVM and RBF_SVM models with BERT word embedding with ISOT Dataset.*

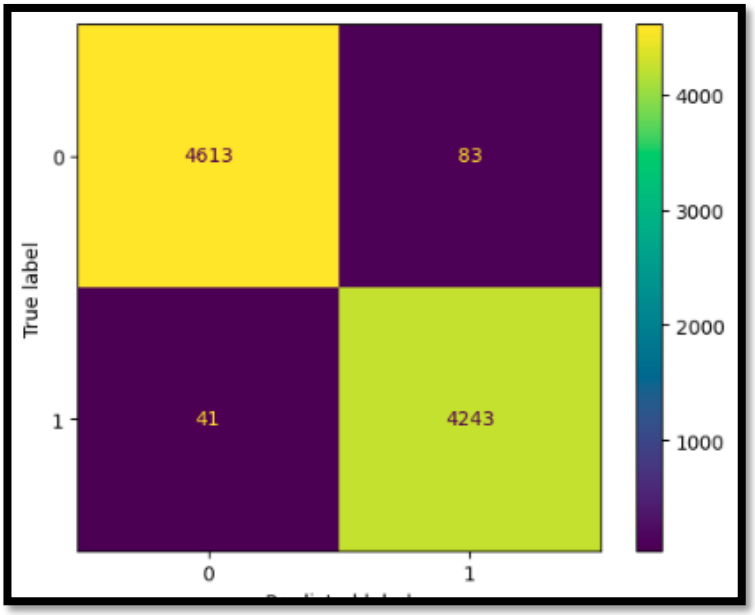


Figure 5.10.*confusion matrix of Logistic Regression model with ISOT Dataset.*

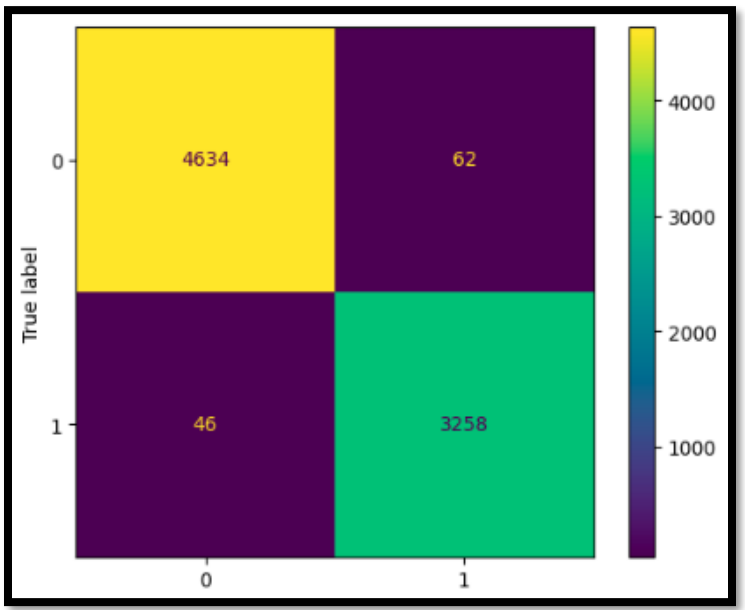


Figure 5.11.*confusion matrix of Random Forest model with ISOT Dataset.*

Figures 5-8, 5-9, 5-10, 5-11 represent the confusion matrixes for the machine learning models we notice that all of them show substantial count in True Positive and True Negative indicating successful predictions for the positive and Negative classes with relatively few misclassifications.

The table **5.1** present the results of suggested models applied to the ISOT Fake News datasets showcasing the evaluation metrics.

Table 5.1.*The performance of suggested models on ISOT Fake News dataset.*

Model	accuracy	recall	Precession	F1
FakeBert	**99.98%**	**100%**	**100%**	**100%**
Lstm	96.71%	97%	97%	97%
Gru	97.14%	97%	97%	97%
Bi-lstm	97.91%	98%	98%	98%
Bi-Gru	99.23%	99%	99%	99%
Bert	99.94%	100%	100%	100%
RoBERTa	**99.98%**	**100%**	**100%**	**100%**
Bert+Linear SVM	99.56%	100%	100%	100%
Bert+rbf SVM	99.65%	100%	100%	100%
Tfidf+ Linear Svm	99. 38%	99%	99%	99%
Tfidf+rbf SVM	98.74%	99%	99%	99%
TFIDF+Logistic Regression	98.57%	99%	99%	99%
Tfidf+Random Forest	98.61%	99%	99%	99%

- **AFND Dataset:**

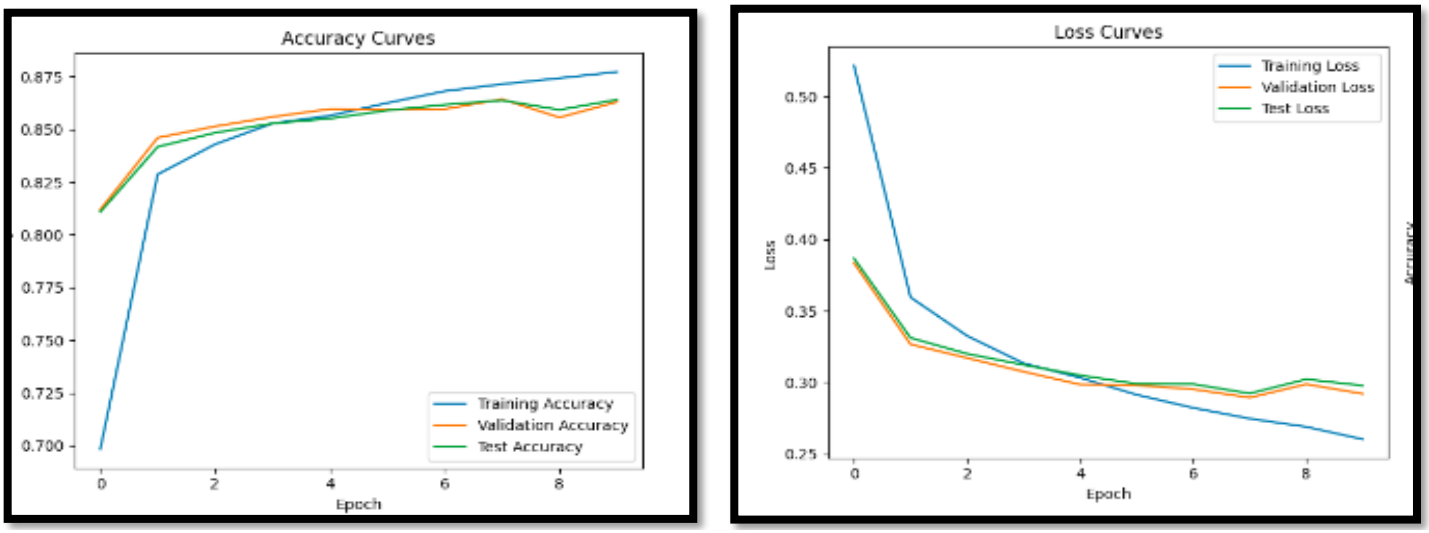


Figure 5.12.*Training and validation accuracy and loss graphs of FakeBert model with AFND Dataset.*

In the provided graphs in Figure 5-12, the training accuracy increases sharply and stabilizes at around 87.72%, while the test and validation accuracy stabilize around 86%. The loss decreases significantly and stabilizes near 25%.

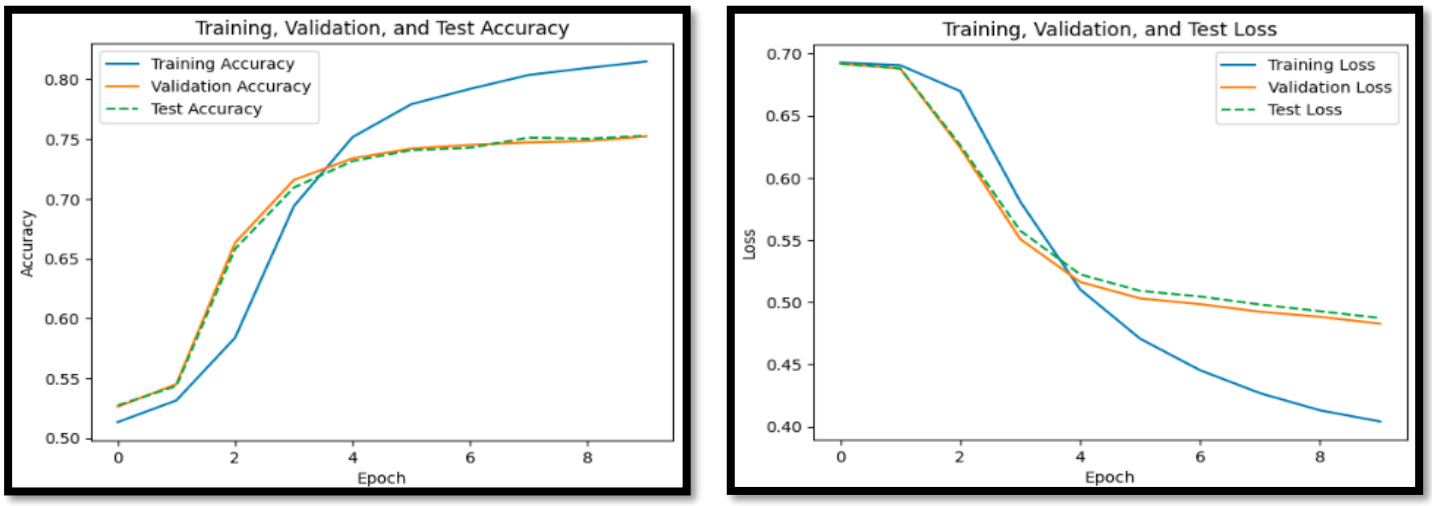


Figure 5.13. *Training and validation accuracy and loss graphs of LSTM model with AFND Dataset.*

In the graphs shown in Figure 5-13, we notice that the accuracy of training is less than the accuracy of testing and verification until the fourth epoch, where the accuracy of training exceed the accuracy of testing and verification, reaching in the end 80%, and we notice the exact opposite for the loss.

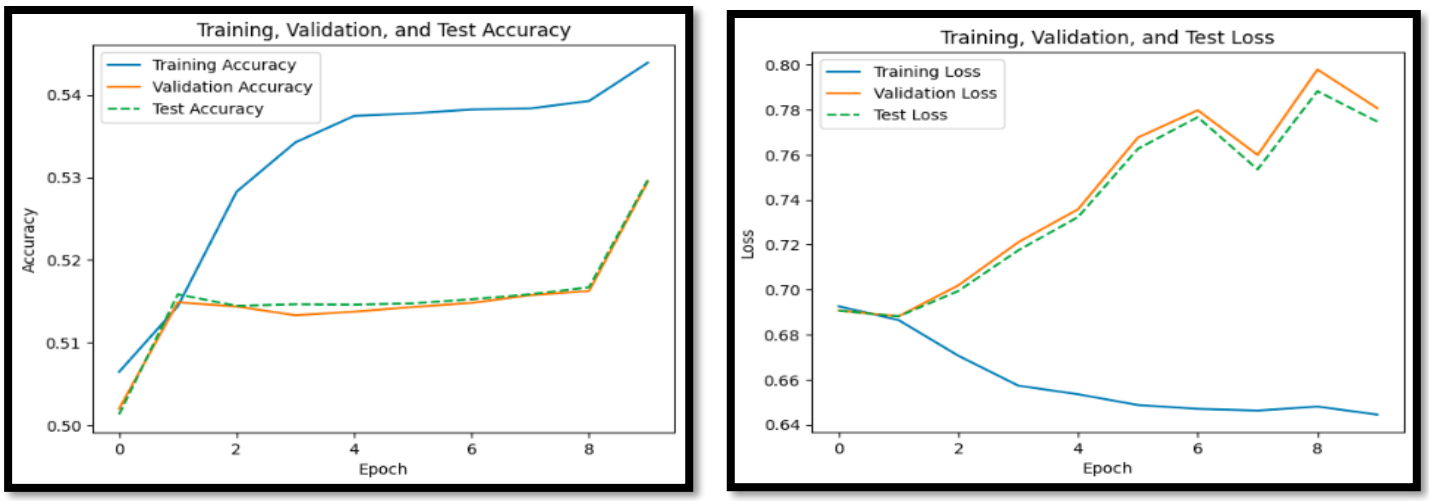


Figure 5.14. *Training and validation accuracy and loss graphs of GRU model with AFND Dataset.*

The graphs in Figure 5-14 display the GRU model curves showing that although training accuracy increases steadily up to 50%, both validation stability and test accuracy diverge after epoch 8 and decrease to a low value, indicating overfitting and There is no improvement in performance. We also notice that the loss value remains high .

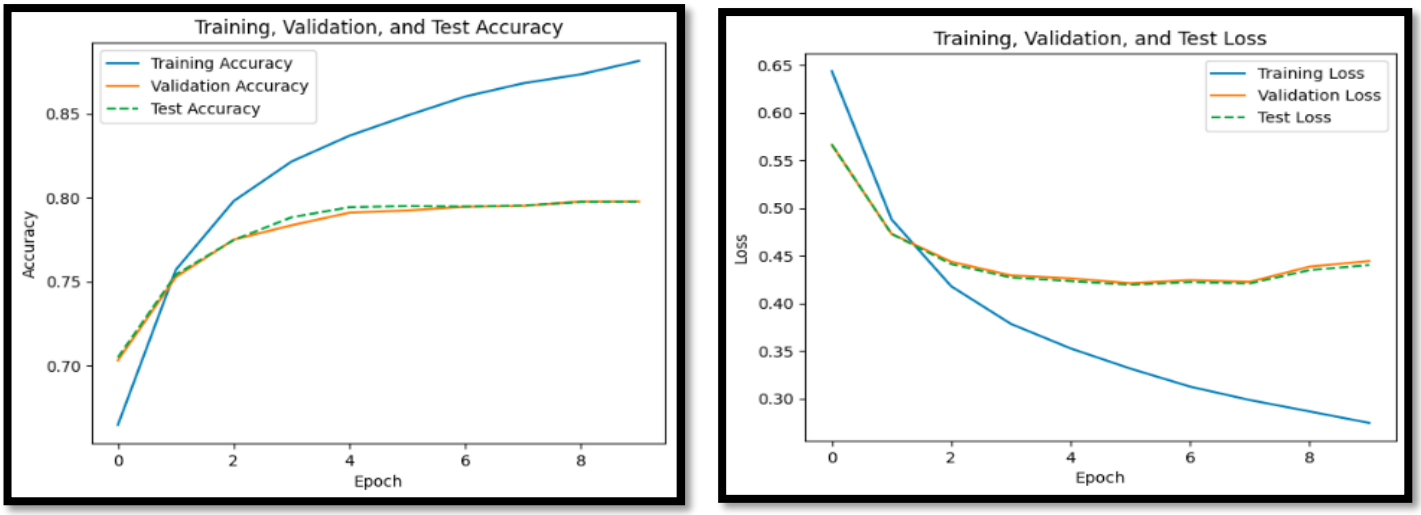


Figure 5.15.*Training and validation accuracy and loss graphs of BILSTM model with AFND Dataset.*

Figure 5-15 displays the accuracy and loss curves for BILSTM, showing that the training accuracy increases steadily to exceed 85%, while the validation and testing accuracies initially rise but stabilize at 80%, indicating overfitting. We observe that the accuracy loss decreases to 30% while the test and verification loss remains high.

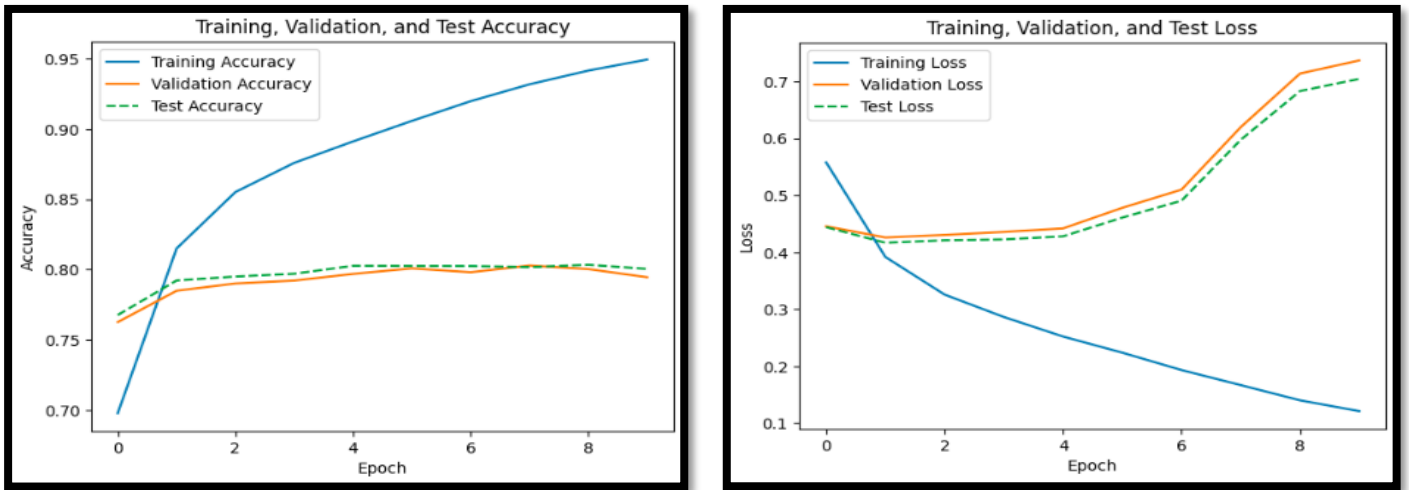


Figure 5.16.*Training and validation accuracy and loss graphs of BIGRU model with AFND Dataset.*

Figure 5-16 shows the accuracy and loss curves for the BIGRU model. The graphs show that training accuracy increases rapidly, reaching 95%, while validation and testing accuracy stabilizes at around 80%. In contrast, we observe that the training loss decreases to 1%, while the testing and validation loss quickly increases to 70%. this suggests that the model is overfitting. The model performs well on the training data but fails to generalize to unseen data.

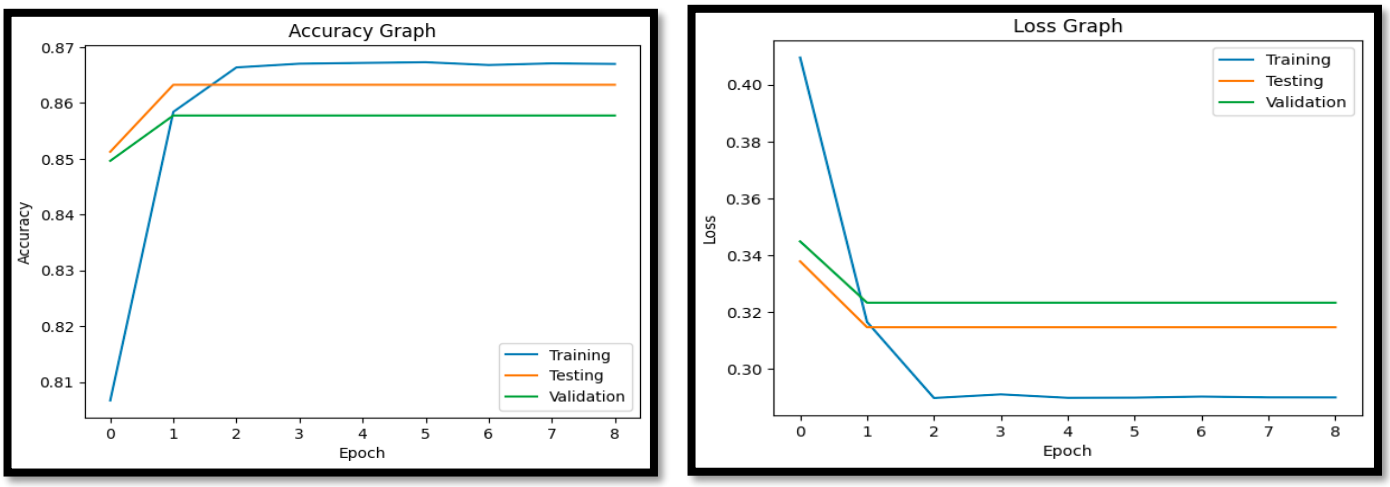


Figure 5.17.*Training and validation accuracy and loss graphs of Bert model with AFND Dataset.*

Figure 5-17 displays the accuracy and loss curves of the BERT model, showing that the training accuracy rises quickly and stabilizes above 0.94, while the validation and test accuracy drops around 0.92, indicating that the model performs well but shows slight overfitting. While the training loss leads to 0.125 and the test and validation loss stabilizes at 0.225.

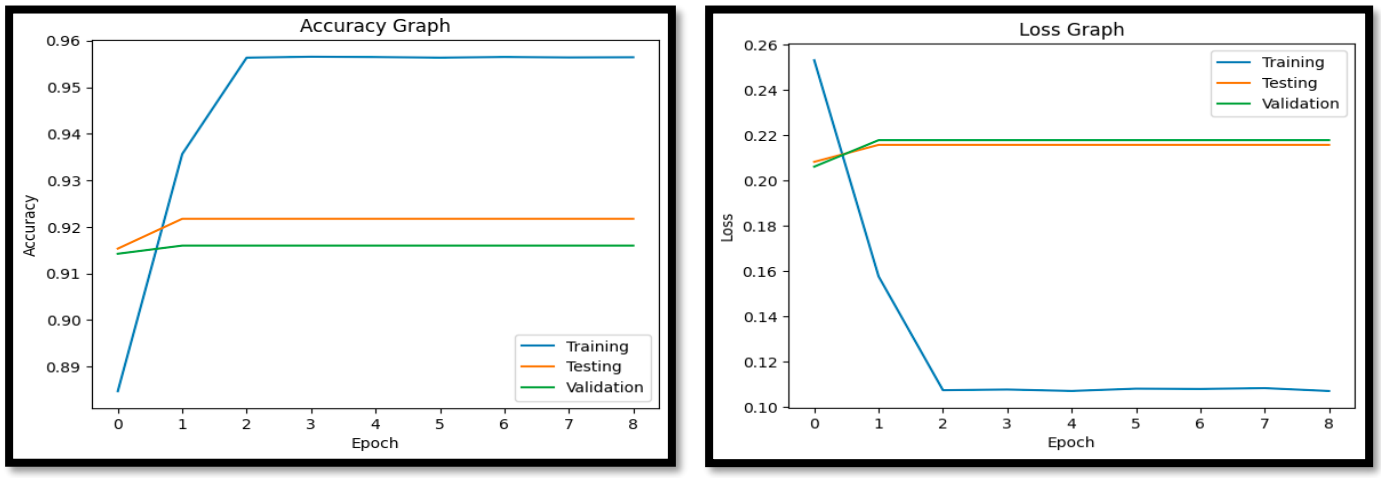


Figure 5.18.*Training and validation accuracy and loss graphs of RoBERTa model with AFND Dataset.*

Figure 5-18 shows the accuracy and loss curves for RoBERTa mode. The graphs show that the training, validation and testing accuracy increases rapidly and stabilizes around 0.87, while the training loss decreased to 0.29 and the testing and validation loss stabilized around 0.31.

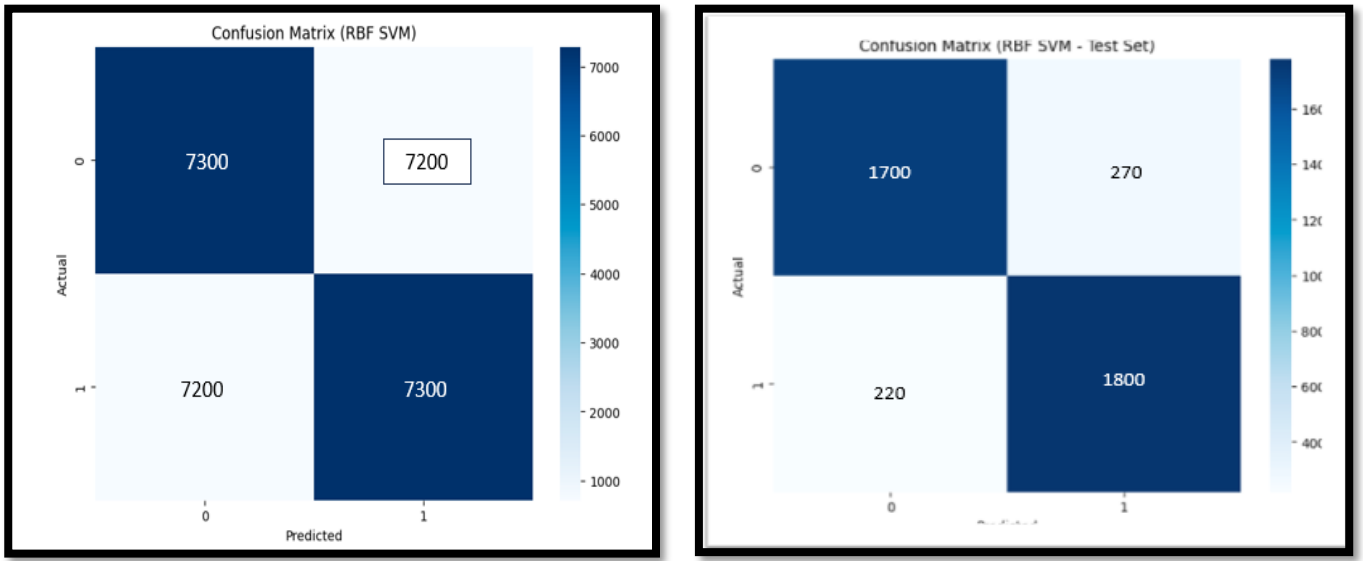


Figure 5.19.*confusion matrix of Linear_SVM anf RBF_SVM models with Bert word embedding with AFND Dataset.*

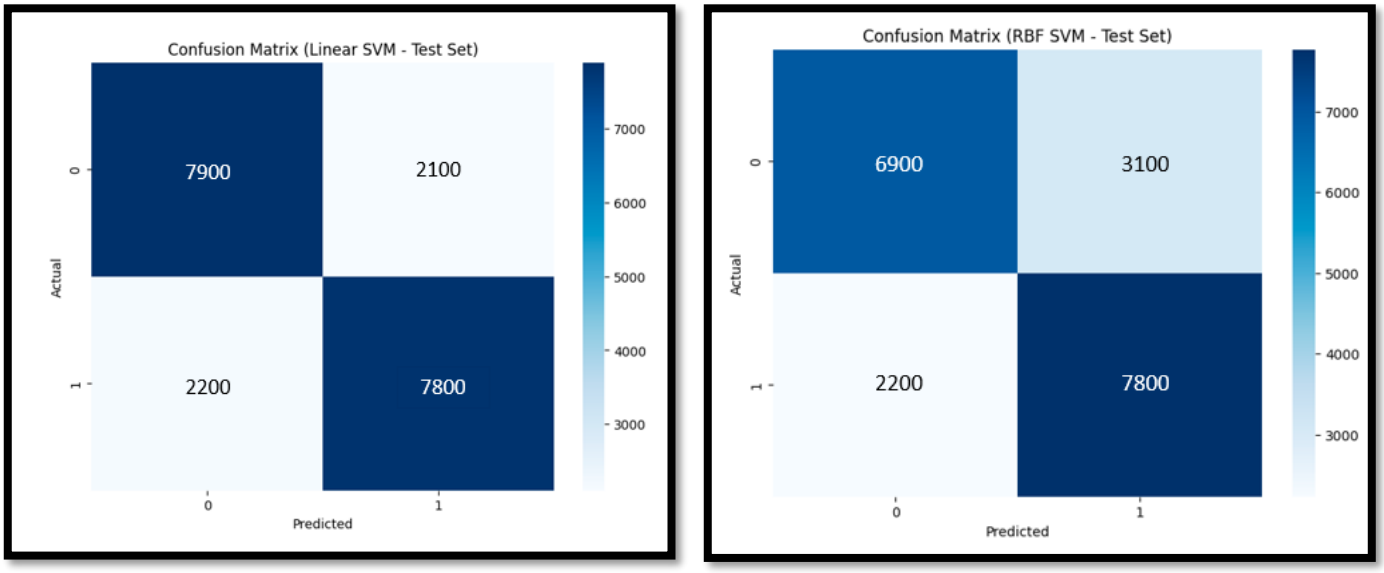


Figure 5.20.*confusion matrix of Linear_SVM and RBF_SVM model with TFIDF word embedding with AFND Dataset*

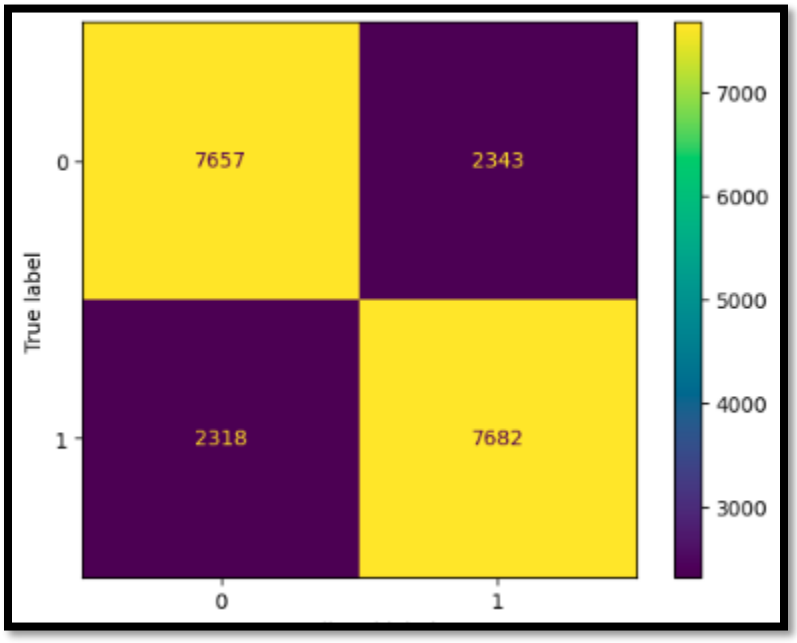


Figure 5.22.*confusion matrix of Logistic Regression with AFND Dataset.*

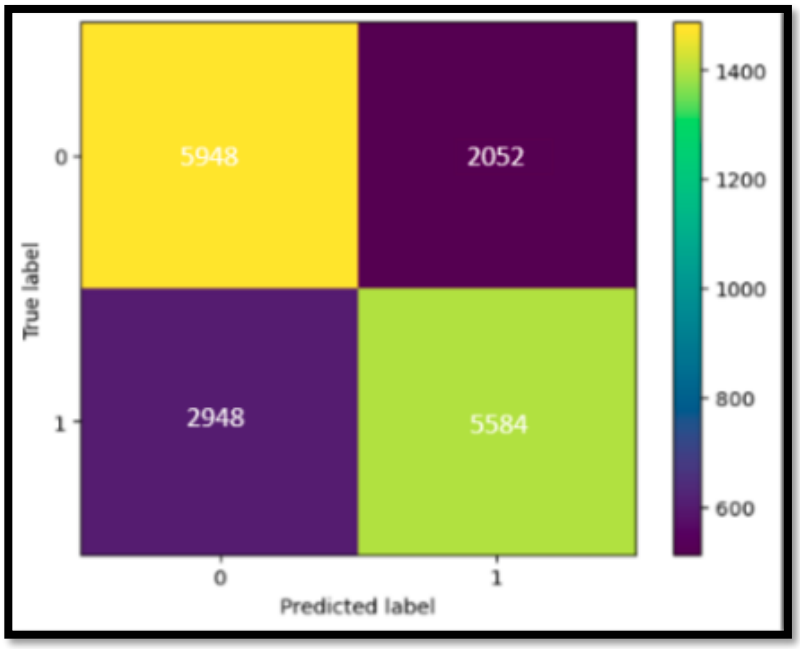


Figure 5.21.*confusion matrix of Random Forest with AFND Dataset.*

Figures 5-19, 5-20, 5-22, 5-23 representing the confusion matrices of the machine learning models with the AFND dataset we note that all of them show a significant number in True Positive and True Negative indicating successful predictions for positive and negative classes with a relatively small number of misclassifications.

The table **5.2** present the results of suggested models applied to the AFND Fake News datasets showcasing the evaluation metrics.

Table 5.2.*The performance of suggested models on AFND Fake News dataset.*

Model	accuracy	recall	Precession	F1
FakeBert	86.40%	86%	87%	86%
Lstm	75.2%	75%	75%	75%
Gru	53.52%	54%	67%	42%
Bi-lstm	79.75%	80%	80%	80%
Bi-Gru	80.30%	80%	80%	80%
Bert	**92.17%**	**92%**	**92%**	**92%**
RoBERTa	86.32%	86%	87%	86%
Bert + rbf SVM	90.99%	91%	91%	91%
Tfidf+ Linear SVM	78.69%	79%	79%	79%
Tfidf+ rbf SVM	73.53%	74%	74%	73%
TFIDF+Logistic Regression	76.69%	77%	77%	77%
Tfidf+Random Forest	72.0%	72%	72%	72%

- **TweepFake dataset:**

The table **5.3** present the results of suggested model applied to the TweepFake datasets showcasing the evaluation metrics.

Table 5.3.*TweepFakes results with each scenario.*

Scenario	accuracy	Recall	Precision	F1
1	**0.6310**	0.7216	0.6113	0.6619
2	0.6232	0.7355	0.6019	0.6620
3	0.6212	0.6771	0.6103	0.6420
4	0.6211	0.6718	0.6113	0.6402
5	0.6233	**0.7409**	0.6010	**0.6637**

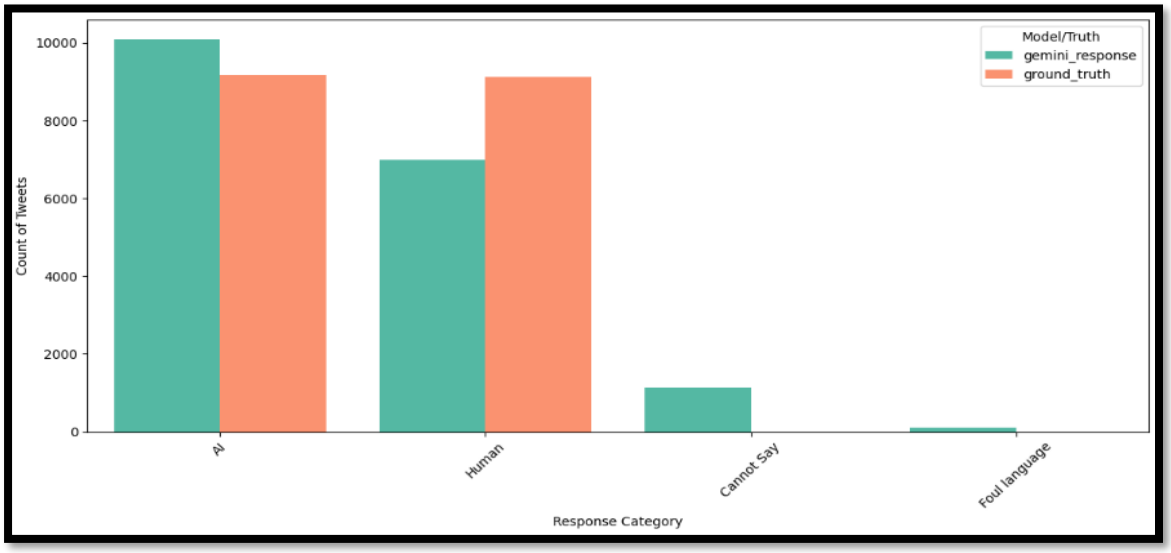


Figure 5.23.*compare between gemini API response and ground truth.*

5.2.2. Comparison between proposed approaches

From our evaluation we noticed that:

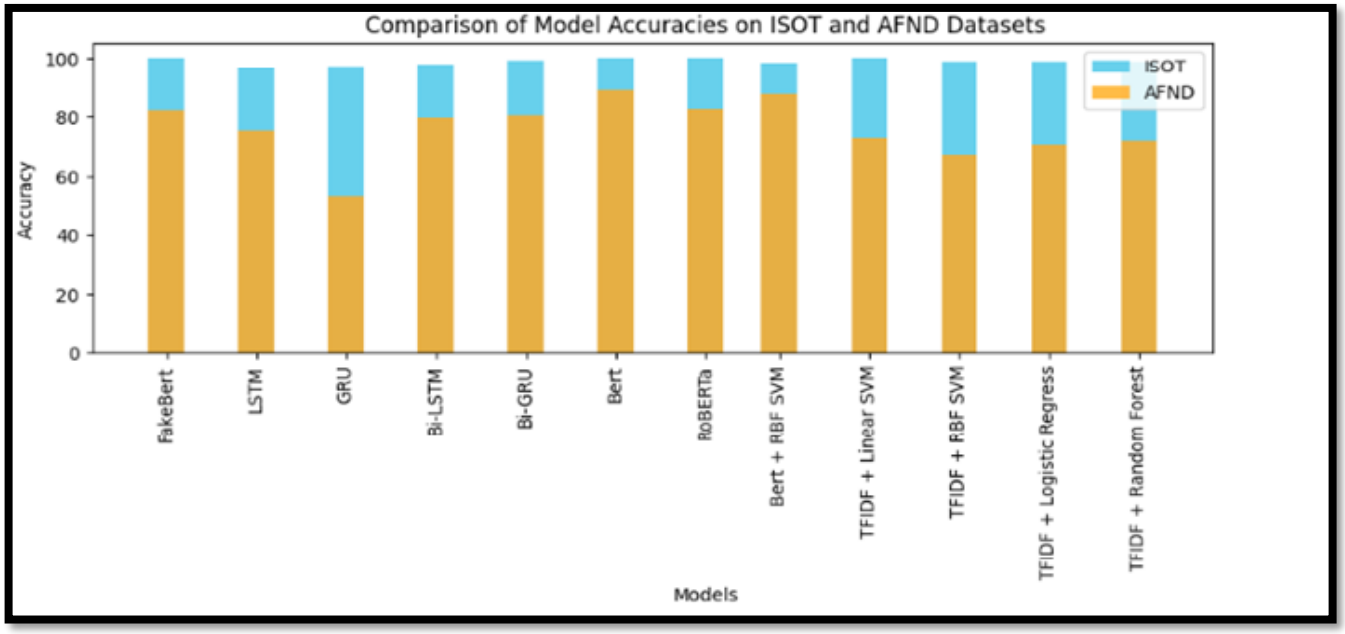


Figure 5.24.*Comparison of Medel Accuracies o ISOT and AFND Datasets.*

- The ISOT Fake News dataset consistently shows higher performance metrics compared to the AFND Fake News dataset across different models. The accuracy of the ISOT dataset remains

consistently high, ranging between 96% and 99.98% for all models. However, for the AFND dataset, the accuracy drops significantly, ranging from as low as 52.96% to a maximum of 92.17%, but does not exceed the accuracy achieved on the ISOT dataset.

- Deep learning models and pre-trained models generally outperform traditional machine learning models on both datasets.

- Combining pre-trained models with deep learning models yields better results. For example, our Fakebert model achieved 99.98% accuracy on ISOT and 86.40% accuracy on AFND. Fine-tuning of RoBERTa and FakeBERT achieved the best accuracy, reaching 99.98% for ISOT while fine-tuning of BERT achieved 92.17% for AFND. Since RoBERTa was not pre-trained on an Arabic-only dataset and we could not use the multilingual XLM RoBERTa due to Our low-capacity devices.

- However, it's worth noting that SVM with BERT word embeddings achieved better results than TF_IDF with SVM on both datasets.

- Bidirectional GRU (Bi-GRU) and Bidirectional LSTM (Bi-LSTM) consistently outperform their unidirectional counterparts (GRU and LSTM) on both datasets. This indicates the importance of capturing contextual information from both directions in sequential data.

- Linear SVM consistently outperforms RBF SVM on both datasets, as indicated by higher accuracy, recall, precision, and F1-score. This suggests that the non-linear decision boundaries provided by RBF kernels are more effective for separating fake news from genuine news.

- Using BERT encoder output directly as input to CNN layers results in better accuracy, up to 99.98%, compared to the method of extracting all features using BERT and then using them in CNN, which resulted in an accuracy of only 73% for same data.

- Our FakeBERT model, utilizing RoBERTa, achieves better accuracy at 99.98%, while the FakeBERT model with BERT, as developed by (Kaliyar et al., 2021) ,achieved an accuracy of 98.90%.

- Table 5-3 shows performance metrics (precision, recall, precision, F1-score) for Gemini API across five scenarios. Overall, the scenarios vary slightly in precision around 0.623, with recall ranging from 0.672 to 0.741. The precision remains stable near 0.611, while the F1 scores range between 0.640 and 0.664, with Scenario 1 having the highest precision, while Scenario 5 has the highest recall and the highest F1 score.

- Due to the low capacities of our machines, we couldn't execute SVM with BERT embeddings on the entire AFND dataset of 100,000 rows. Instead, we executed it on a subset of 20,000 rows and achieved an accuracy of 81.95% for Linear SVM and 87.67% for the non-linear SVM.

5.2.3. Comparison of proposed approach with existing works

This part is focused on evaluating our models by comparing them to previous studies in the literature that utilized the ISOT, AFND and TweepFake datasets for both training and testing purposes.

Table 5.4 presents the comparative analysis of the present work with existing works.

Table 5.4. *Comparison of proposed approach with previous contributions results.*

Ref	Dataset	Accuracy
(Fagni et al., 2021)	TweepFake	GPT-2= 90%
Proposed approach Gemini API		0.6310
(Sastrawan et al., 2022)	Isot	CNN = 99.88%, Bi-LSTM = 99.95%, ResNet = 99.90%
Proposed FakeBert		**99.98%**
Proposed BILSTM		97.91%
Proposed BERT		99.94%
(Sorour et al., 2022)	AFND	CNN-LSTM= 0.81, LSTM=0.67 AraBERTv02=0.68
Proposed FakeBert		86.40%
Proposed LSTM		75.2%
Proposed Bert		**92.17%**

- **Discussion**

The model proposed GPT-2 by (Fagni et al., 2021) achieves 90% accuracy superior to our Gemini model.

The proposed BILST by (Sastrawan et al., 2022) achieves accuracy superior than our BILSTM proposed model while its other models were slightly lower than the accuracy of our proposed models.

For the AFND dataset, our models outperform those of (Sorour et al. (2022).

5.3. Web Application To detect Fake News

After training our models with ISOT and AFND datasets, we selected the BERT model that achieved the best results for building our application by saving it as a pickle file.

5.3.1 Build the application

- **Streamlit** [15]

Streamlit is a free and open-source framework that enables the quick and easy creation of beautiful web applications for machine learning and data science projects using straightforward Python scripts. With support for hot-reloading, the application updates live as the engineer edits and saves their file. Building an app with Streamlit requires just a few lines of code using the Streamlit API. Adding widgets is as simple as declaring variables. There is no need to write a backend, define routes, or handle HTTP requests. Streamlit is also easy to deploy and manage. [xii]

- **Deploy the model**

Deploying a machine learning or a deep learning model, also known as model deployment, involves integrating a trained model into a production environment where it can accept input data and generate predictions. The primary goal of model deployment is to make the model's predictions accessible to users, management, or other systems. This process ensures that the insights derived from the model can be utilized effectively in real-world applications.

Model deployment is a critical aspect of machine learning and a deep learning systems architecture, which encompasses the design and interaction of software components to achieve

[15] https://www.streamlit.io/

specific objectives. Effective deployment requires considering various factors such as scalability, reliability, latency, and integration with existing systems.[xiii]

We created two interfaces one for the Arabic news and other for English news



Figure 5.25*.Detect Arabic Fake News.*



Figure 5.26.*Detect English Fake News.*

5.4. Conclusion

This chapter discusses the outcomes of our methodology used to address the issue of fake news and deepfakes. Additionally, we discovered that leveraging pre-trained for word embeddings and combining them with classifiers can improve predictive capabilities. Our findings suggest that pre-trained classifiers achieve better results compared to both deep learning and traditional machine learning models, The key to improving the detection of automated fake news is having access to top-notch training data and pre-existing models that perform exceptionally well.

General Conclusion

In this study, we address the major challenge of detecting deepfakes and fake news on social media. We have identified different concepts and classifications of fake news, explored the factors that contribute to its viral nature, and considered its broad impacts on individuals and society, especially in critical areas such as health. We reviewed previous detection techniques and highlighted the challenges associated with them.

Our research on using machine learning, deep learning, and natural language processing (NLP) to detect fake news and deep fake content has yielded several important findings. We examined models such as FakeBERT, LSTM, Bi-LSTM, GRU, Bi-GRU, BERT, RoBERTa, Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF) and Gemini API with binary classification. In addition, we coded the data and coded sentences for analysis.

Our evaluation metrics—accuracy, precision, recall, and F1 score— show that deep learning and pre-trained models generally outperform traditional machine learning models. Notably, the ISOT Fake News dataset consistently produced higher performance metrics compared to the AFND dataset. For example, our FakeBERT model and Fine-tuning of RoBERTa yielded the best results, with 99.98% accuracy for ISOT while Fine-tuning BERT achieved the best accuracy on AFND dataset with 92.17%, confirming the value of using pre-trained representations for fake news detection.

Using the Gemini API with binary classification by five scenarios resulted in little variation in precision around 0.623, with recall ranging from 0.672 to 0.741. Precision also remains stable near 0.611, while F1 scores range between 0.640 and 0.664. It is worth noting that Scenario 1 shows the highest precision, while Scenario 5 achieves the highest recall and F1 score.

The superior performance of bidirectional LSTM (Bi-LSTM) and bidirectional GRU (Bi-GRU) over their unidirectional counterparts highlights the importance of capturing contextual information from both directions in sequential data. In addition, using BERT encoder outputs directly as input to CNN layers resulted in higher accuracy compared to other methods.

Our results indicate that pre-trained classifiers outperform traditional deep learning and machine learning models. The key to enhancing fake news detection is access to high-quality training data and pre-existing, highly effective models.

Although significant progress has been made in detecting fake news and deepfakes, innovation and continuous improvement are necessary to effectively address these evolving threats. Future research should focus on continued development in this vital area to keep pace with the changing misinformation landscape.

For the future works we will work on Arabic datasets and Large Language Models (LLMs) for Fake News Detection and AI Generation. We will work to create new approaches for Enhanced the Performance. and develop an android application for Real-Time Fake News Detection.

Bibliography

Abbasi, A., Zhang, Z., Zimbra, D., Chen, H., & Nunamaker Jr, J. F. (2010). Detecting fake websites: The contribution of statistical learning theory. *Mis Quarterly*, 435–461.

Abramson, J. B., Orren, G. R., & Arterton, F. C. (1990). *Electronic Commonwealth: The Impact of New Media Technologies on Democratic Politics*. Basic Books, Inc.

Ahmed, H., Traore, I., & Saad, S. (2017). Detection of online fake news using n-gram analysis and machine learning techniques. *Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments: First International Conference, ISDDC 2017, Vancouver, BC, Canada, October 26-28, 2017, Proceedings 1*, 127–138.

Ahmed, H., Traore, I., & Saad, S. (2018). Detecting opinion spams and fake news using text classification. *Security and Privacy*, *1*(1), e9.

Al-Yahya, M., Al-Khalifa, H., Al-Baity, H., AlSaeed, D., & Essam, A. (2021). Arabic fake news detection: comparative study of neural networks and transformer-based approaches. *Complexity*, *2021*, 1–10.

Ali, I., Ayub, M. N. Bin, Shivakumara, P., & Noor, N. F. B. M. (2022). Fake news detection techniques on social media: A survey. *Wireless Communications and Mobile Computing*, *2022*.

Ali, S., DiPaola, D., & Breazeal, C. (2021). What are GANs?: Introducing generative adversarial networks to middle school students. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(17), 15472–15479.

Alkhatib, K., Najadat, H., Hmeidi, I., & Shatnawi, M. K. A. (2013). Stock price prediction using k-nearest neighbor (kNN) algorithm. *International Journal of Business, Humanities and Technology*, *3*(3), 32–44.

Allcott, H., & Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, *31*(2), 211–236.

Alyoubi, S., Kalkatawi, M., & Abukhodair, F. (2023). The detection of fake news in arabic tweets using deep learning. *Applied Sciences*, *13*(14), 8209.

Apuke, O. D., & Omar, B. (2021). Fake news and COVID-19: modelling the predictors of fake news sharing among social media users. *Telematics and Informatics*, *56*, 101475.

Arun, C. (2022). On WhatsApp, Rumours and Lynchings. *Acts of Media: Law and Media in Contemporary India*, *2022*, 122–140.

Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M., & Szlam, A. (2019). Real or fake? learning to discriminate machine from human generated text. *ArXiv Preprint ArXiv:1906.03351*.

Baum, M. A., & Lazer, D. (2017). Google and Facebook Aren't Fighting Fake News with the Right Weapons. *Los Angeles Times, May*, *8*.

Bhavani, A., & Kumar, B. S. (2021). A review of state art of text classification algorithms. *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 1484–1490.

Bhuvaneswari, A., Thomas, J. T. J., & Kesavan, P. (2019). Embedded bi-directional GRU and LSTMLearning models to predict disasterson twitter data. *Procedia Computer Science*, *165*, 511–516.

Bishop, M. (2019). Healthcare social media for consumer informatics. *Consumer Informatics and Digital*

Health: Solutions for Health and Health Care, 61–86.

Blackledge, C., & Atapour-Abarghouei, A. (2021). Transforming fake news: Robust generalisable news classification using transformers. *2021 IEEE International Conference on Big Data (Big Data)*, 3960–3968.

Brodie, I. (2018). Pretend news, false news, fake news: The onion as put-on, prank, and legend. *Journal of American Folklore*, *131*(522), 451–459.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., & Askell, A. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901.

Brunvand, J. H. (1963). A classification for shaggy dog stories. *The Journal of American Folklore*, *76*(299), 42–68.

Burkhardt, J. M. (2017). *Combating fake news in the digital age* (Vol. 53, Issue 8). American Library Association Chicago, IL, USA.

CDC. (2019). *Measles Cases and Outbreaks, Centre for Disease Control and Prevention*.

Chauhan, N. K., & Singh, K. (2018). A review on conventional machine learning vs deep learning. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 347–352.

Chen, Y., Conroy, N. J., & Rubin, V. L. (2015). Misleading online content: recognizing clickbait as" false news". *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, 15–19.

Cheng, S. (2023). *Recurrent neural networks*.

Choudhury, D., & Acharjee, T. (2023). A novel approach to fake news detection in social networks using genetic algorithm applying machine learning classifiers. *Multimedia Tools and Applications*, *82*(6), 9029–9045.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv Preprint ArXiv:1412.3555*.

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

Edgar, A., & Sedgwick, P. (2007). *Cultural theory: The key concepts*. Routledge.

Fagni, T., Falchi, F., Gambini, M., Martella, A., & Tesconi, M. (2021). TweepFake: About detecting deepfake tweets. *Plos One*, *16*(5), e0251415.

Fahy, E., Hardikar, R., Fox, A., & Mackay, S. (2014). Quality of patient health information on the Internet: reviewing a complex and evolving landscape. *The Australasian Medical Journal*, *7*(1), 24.

Faris, H., Aljarah, I., & Mirjalili, S. (2016). Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Applied Intelligence*, *45*, 322–332.

Fetzer, J. H. (2004). Information: Does it have to be true? *Minds and Machines*, *14*, 223–229.

Figueira, Á., & Oliveira, L. (2017). The current state of fake news: challenges and opportunities. *Procedia Computer Science*, *121*, 817–825.

Fouad, K. M., Sabbeh, S. F., & Medhat, W. (2022). Arabic Fake News Detection Using Deep Learning. *Computers, Materials & Continua*, *71*(2).

Fox, S., & Duggan, M. (2013). Health online 2013. *Health*, *2013*, 1–55.

Fu, L. Y., Zook, K., Spoehr-Labutta, Z., Hu, P., & Joseph, J. G. (2016). Search engine ranking, quality, and content of web pages that are critical versus noncritical of human papillomavirus vaccine. *Journal of Adolescent Health*, *58*(1), 33–39.

Gallé, M., Rozen, J., Kruszewski, G., & Elsahar, H. (2021). Unsupervised and distributional detection of machine-generated text. *ArXiv Preprint ArXiv:2111.02878*.

Gelfert, A. (2018). Fake news: A definition. *Informal Logic*, *38*(1), 84–117.

Ghorbanpour, F., Ramezani, M., Fazli, M. A., & Rabiee, H. R. (2023). FNR: a similarity and transformer-based approach to detect multi-modal fake news in social media. *Social Network Analysis and Mining*, *13*(1), 56.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Guimarães, N., Figueira, Á., & Torgo, L. (2021). Can fake news detection models maintain the performance through time? A longitudinal evaluation of twitter publications. *Mathematics*, *9*(22), 2988.

Gurney, K. (2018). *An introduction to neural networks*. CRC press.

Hangloo, S., & Arora, B. (2022). Combating multimodal fake news on social media: methods, datasets, and future perspective. *Multimedia Systems*, *28*(6), 2391–2422.

Harrag, F., & Djahli, M. K. (2022). Arabic fake news detection: A fact checking based deep learning approach. *Transactions on Asian and Low-Resource Language Information Processing*, *21*(4), 1–34.

Hern, A. (2017). Russian troll factories: researchers damn Twitter's refusal to share data. *The Guardian*.

Himdi, H., Weir, G., Assiri, F., & Al-Barhamtoshy, H. (2022). Arabic fake news detection based on textual analysis. *Arabian Journal for Science and Engineering*, *47*(8), 10453–10469.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261–266.

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

Hoi SCH, Jin R, L. M. R. (2006). *Large-scale text categorization by batch mode learning". In: Proc 15th Int World Wide Web conference (WWW2006), Edinburgh, England, UK,*.

Jadhav, S. S., & Thepade, S. D. (2019). Fake news identification and classification using DSSM and improved recurrent neural network classifier. *Applied Artificial Intelligence*, *33*(12), 1058–1068.

Jain, A., & Kasbe, A. (2018). Fake news detection. *2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 1–5.

Jiang, T. A. O., Li, J. P., Haq, A. U., Saboor, A., & Ali, A. (2021). A novel stacking approach for accurate detection of fake news. *IEEE Access*, *9*, 22626–22639.

Kaliyar, R. K., Goswami, A., & Narang, P. (2021). FakeBERT: Fake news detection in social media with a BERT-

based deep learning approach. *Multimedia Tools and Applications*, *80*(8), 11765–11788.

Karim, M. R., Dey, S. K., Islam, T., Sarker, S., Menon, M. H., Hossain, K., Hossain, M. A., & Decker, S. (2021). Deephateexplainer: Explainable hate speech detection in under-resourced bengali language. *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10.

Khalil, A., Jarrah, M., Aldwairi, M., & Jaradat, M. (2022). AFND: Arabic fake news dataset for the detection and classification of articles credibility. *Data in Brief*, *42*, 108141.

Khan, S., Hakak, S., Deepa, N., Prabadevi, B., Dev, K., & Trelova, S. (2022). Detecting COVID-19-related fake news using feature extraction. *Frontiers in Public Health*, *9*, 788074.

Kovach, B., & Rosenstiel, T. (2007). What newspeople should know and the public should expect. *The Elements of Journalism*.

Kshetri, N., & Voas, J. (2017). The economics of "fake news." *IT Professional*, *19*(6), 8–12.

Kumar, S., Asthana, R., Upadhyay, S., Upreti, N., & Akbar, M. (2020). Fake news detection using deep learning models: A novel approach. *Transactions on Emerging Telecommunications Technologies*, *31*(2), e3767.

Ling, W., Dyer, C., Black, A. W., & Trancoso, I. (2015). Two/too simple adaptations of word2vec for syntax problems. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1299–1304.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv Preprint ArXiv:1907.11692*.

M. Schuster and K. K. Paliwal. (1997). Bidirectional recurrent neural networks. *IEEE Trans*, *45*, 2673– 2681.

Ma, J., Gao, W., & Wong, K.-F. (2017). *Detect rumors in microblog posts using propagation structure via kernel learning*.

MacDougall, C. D. (1958). *Hoaxes, vol. 465*. Dover Publications.

Mahir, E. M., Akhter, S., & Huq, M. R. (2019). Detecting fake news using machine learning and deep learning algorithms. *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, 1–5.

Mandical, R. R., Mamatha, N., Shivakumar, N., Monica, R., & Krishna, A. N. (2020). Identification of fake news using machine learning. *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 1–6.

Manish, M. K. S., Jawed, J. A., Alam, M. A. A., Kamlesh, K. K. R., & Sachin, S. K. (2022). *A comparative study of computational fake news detection on social media*.

Masood, A. D., Abdulazeez, A. M., & Zeebaree, D. Q. (2020). Machine learning supervised algorithms of gene selection: A review. *Machine Learning*, *62*, 3.

Mavridis, G. (2018). *Fake news and social media: How Greek users identify and curb misinformation online*. Malmö universitet/Kultur och samhälle.

Mazari, A. C., Aliane, H., & Alimazighi, Z. (2013, May). A conceptual indexing approach for Arabic texts. In *2013 ACS International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-1). IEEE.

Mazari, A. C., & Djeffal, A. (2021). A combined method of naïve-bayes and pooling strategy for building test collection for Arabic/English information retrieval. International Journal of Computing and Digital Systems, 10(1), 629–638. https://doi.org/10.12785/IJCDS/100161

Mazari, A. C., & Djeffal, A. (2021, February). Deep Learning-Based Sentiment Analysis of Algerian Dialect during Hirak 2019. In *2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH)* (pp. 233-236). IEEE.

Mazari, A. C., & Djeffal, A. (2022). Sentiment Analysis of Algerian Dialect Using Machine Learning and Deep Learning with Word2vec. Informatica, 46(6), 67–78. https://doi.org/10.31449/INF.V46I6.3340

Mazari, A. C., & Djeffal, A. (2022). Hybrid Query Expansion Model Based on Pseudo Relevance Feedback and Semantic Tree for Arabic IR. *International Journal of Information Retrieval Research (IJIRR)*, *12*(1), 1-16.

Mazari, A. C., & Kheddar, H. (2023). Deep Learning-based Analysis of Algerian Dialect Dataset Targeted Hate Speech, Offensive Language and Cyberbullying. *International Journal of Computing and Digital Systems*.

Mazari, A. C., Boudoukhani, N., & Djeffal, A. (2024). BERT-based ensemble learning for multi-aspect hate speech detection. *Cluster Computing*, *27*(1), 325-339. https://doi.org/10.1007/s10586-022-03956-x

Nan, Q., Cao, J., Zhu, Y., Wang, Y., & Li, J. (2021). MDFEND: Multi-domain fake news detection. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3343–3347.

Narayanan, V., Howard, P. N., Kollanyi, B., & Elswah, M. (2017). Russian involvement and junk news during Brexit. *The Computational Propaganda Project. Algorithms, Automation and Digital Politics. Https://Comprop. Oii. Ox. Ac. Uk/Research/Working-Papers/Russia-and-Brexit*.

Nguyen-Son, H.-Q., Tieu, N.-D. T., Nguyen, H. H., Yamagishi, J., & Zen, I. E. (2017). Identifying computer-generated text using statistical analysis. *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 1504–1511.

Orabi, M., Mouheb, D., Al Aghbari, Z., & Kamel, I. (2020). Detection of bots in social media: a systematic review. *Information Processing & Management*, *57*(4), 102250.

Oshikawa, R., Qian, J., & Wang, W. Y. (2018). A survey on natural language processing for fake news detection. *ArXiv Preprint ArXiv:1811.00770*.

Ozbay, F. A., & Alatas, B. (2020). Fake news detection within online social media using supervised artificial intelligence algorithms. *Physica A: Statistical Mechanics and Its Applications*, *540*, 123174.

Padnekar, S. M., Kumar, G. S., & Deepak, P. (2020). Bilstm-autoencoder architecture for stance prediction. *2020 International Conference on Data Science and Engineering (ICDSE)*, 1–5.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217–222.

Pasquale, F. (2015). *The black box society: The secret algorithms that control money and information*. Harvard University Press.

Payne, A. (2017). Russia used a network of 150,000 Twitter accounts to meddle in Brexit. *Business Insider*.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation.

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543.

Preston, S., Anderson, A., Robertson, D. J., Shephard, M. P., & Huhe, N. (2021). Detecting fake news on Facebook: The role of emotional intelligence. *Plos One*, *16*(3), e0246757.

Rachid, B. A., Azza, H., & Ghezala, H. H. B. (2020, July). Classification of cyberbullying text in Arabic. In *2020 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-7). IEEE.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Reilly, I. (2018). F for Fake: Propaganda! Hoaxing! Hacking! Partisanship! and Activism! in the fake news ecology. *The Journal of American Culture*, *41*(2), 139–152.

Revez, J., & Corujo, L. (2021). Librarians against fake news: A systematic literature review of library practices (Jan. 2018–Sept. 2020). *The Journal of Academic Librarianship*, *47*(2), 102304.

Rusli, A., Young, J. C., & Iswari, N. M. S. (2020). Identifying fake news in Indonesian via supervised binary text classification. *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, 86–90.

Safieddine, F., & Ibrahim, Y. (2020). History of fake news. *Fake News in an Era of Social Media: Tracking Viral Contagion, Rowland and Littlefield*, 1.

Sastrawan, I. K., Bayupati, I. P. A., & Arsa, D. M. S. (2022). Detection of fake news using deep learning CNN–RNN based methods. ICT Express, 8(3), 396–408. https://doi.org/10.1016/J.ICTE.2021.10.003

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Schwitzer, G. (2017). Pollution of health news. In *BMJ* (Vol. 356). British Medical Journal Publishing Group.

Shao, C., Hui, P.-M., Wang, L., Jiang, X., Flammini, A., Menczer, F., & Ciampaglia, G. L. (2018). Anatomy of an online misinformation network. *Plos One*, *13*(4), e0196087.

Shoemaker, P. J., & Reese, S. D. (2013). *Mediating the message in the 21st century: A media sociology perspective*. Routledge.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2020). Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data*, *8*(3), 171–188.

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, *19*(1), 22–36.

Shu, K., Wang, S., Lee, D., & Liu, H. (2020). Mining disinformation and fake news: Concepts, methods, and recent advancements. *Disinformation, Misinformation, and Fake News in Social Media: Emerging Research Challenges and Opportunities*, 1–19.

Singh, A., Kumar, A., & Akhtar, Z. (2021). Bitcoin Price Prediction: A Deep Learning Approach. *2021 8th*

International Conference on Signal Processing and Integrated Networks (SPIN), 1053–1058.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., & Kreps, S. (2019). Release strategies and the social impacts of language models. *ArXiv Preprint ArXiv:1908.09203*.

Sorour, S. E., & Abdelkader, H. E. (2022). AFND: Arabic fake news detection with an ensemble deep CNN-LSTM model. *J. Theor. Appl. Inf. Technol*, *100*(14), 5072-5086.

Spenkuch, J. L., & Toniatti, D. (2016). Political advertising and election outcomes. *Kilts Center for Marketing at Chicago Booth–Nielsen Dataset Paper Series*, 1–46.

Subramanian, S. (2017). Inside the Macedonian fake-news complex. *Wired Magazine*, *15*.

Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., & De Alfaro, L. (2017). Some like it hoax: Automated fake news detection in social networks. *ArXiv Preprint ArXiv:1704.07506*.

Tambuscio, M., Ruffo, G., Flammini, A., & Menczer, F. (2015). Fact-checking effect on viral hoaxes: A model of misinformation spread in social networks. *Proceedings of the 24th International Conference on World Wide Web*, 977–982.

Tewksbury, D. (2003). What do Americans really want to know? Tracking the behavior of news readers on the Internet. *Journal of Communication*, *53*(4), 694–710.

Thota, A., Tilak, P., Ahluwalia, S., & Lohia, N. (2018). Fake news detection: a deep learning approach. *SMU Data Science Review*, *1*(3), 10.

Townsend, T. (2016). Meet the Romanian Trump Fan behind a major fake news site. *Recuperado de: Http://Www. Inc. Com/Tess-Townsend/Ending-Fedtrump-Facebook. Html*.

Varma, R., Verma, Y., Vijayvargiya, P., & Churi, P. P. (2021). A systematic survey on deep learning and machine learning approaches of fake news detection in the pre-and post-COVID-19 pandemic. *International Journal of Intelligent Computing and Cybernetics*, *14*(4), 617–646.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Verma, A., Mittal, V., & Dawn, S. (2019). FIND: Fake information and news detections using deep learning. *2019 Twelfth International Conference on Contemporary Computing (IC3)*, 1–7.

Volkova, S., Shaffer, K., Jang, J. Y., & Hodas, N. (2017). Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 647–653.

Wardle, C. (2017). Fake news. It's complicated. *First Draft*, *16*, 1–11.

Wardle, C., & Derakhshan, H. (2018). Thinking about 'information disorder': formats of misinformation, disinformation, and mal-information. *Journalism,'Fake News'& Disinformation*, 43–54.

Waszak, P. M., Kasprzycka-Waszak, W., & Kubanek, A. (2018). The spread of medical fake news in social media–the pilot quantitative study. *Health Policy and Technology*, *7*(2), 115–118.

Woolley, S. C., & Howard, P. N. (2018). *Computational propaganda: Political parties, politicians, and political manipulation on social media*. Oxford University Press.

Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z., & Yu, P. S. (2018). TI-CNN: Convolutional neural networks for fake news detection. *ArXiv Preprint ArXiv:1806.00749*.

Yonghui Wu et al.. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*.

Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management*, *57*(2), 102025.

Web Links

[i] https://www.nytimes.com/by/david-leonhardt/ Retrieved: 02/01/2024

[ii] https://spectrum.ieee.org/what-is-deepfake/ Retrieved: 02/02/2024

[iii] https://insights.som.yale.edu/insights/does-fake-news-sway-financial-markets/ 02/2024

[iv] https://huggingface.co/docs/transformers/model_doc/bert/ Retrieved: 06/20/2024

[v] https://www.geeksforgeeks.org/machine-learning/ Retrieved: 02/2024

[vi] https://paperswithcode.com/dataset/liar/ Retrieved: 27/05/2024

[vii] https://paperswithcode.com/dataset/fakenewsnet/ Retrieved: 27/05/2024

[viii] https://paperswithcode.com/dataset/fnc-1/ Retrieved: 27/25/2024

[ix] https://paperswithcode.com/dataset/weibo21/ Retrieved: 27/05/2024

[x] https://www.ibm.com/topics/confusion-matrix/ Retrieved: 03/2024

[xi] https://ai.google.dev/gemini-api/docs/get-started/tutorial?hl=fr&lang=python/ 22/05/2024

[xii] https://www.datacamp.com/tutorial/streamlit/ Retrieved: 05/2024

[xiii] https://builtin.com/machine-learning/model-deployment/ Retrieved: 05/2024