NewsTrueBytes

Report submitted in partial fulfillment of the requirement for the degree of

B.Tech
in
Computer Science & Engineering

Under the supervision
of
Dr. Shweta Taneja
Associate Professor, CSE

by

G-33

Khushi Kulshreshtha (07020802721)
Vibhuti Gupta (05120802721)
Priyanshu Gautam (02320802721)
Shivansh Sharma (36220802721)

Department of Computer Science & Engineering
Bhagwan Parshuram Institute of Technology
PSP-4, Sec-17, Rohini, Delhi-89

OCTOBER-2024

Table of Contents

List of Figures

List of Tables

DECLARATION

-

This is to certify that Report titled “NewsTrueBytes”, is submitted by us in partial fulfillment of the requirement for the award of degree B.Tech. in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. It comprises of our original work. The due acknowledgement has been made in the report for using others work.

25 October 2024

Khushi Kulshreshtha, 07020802721

Vibhuti Gupta, 05120802721

Priyanshu Gautam, 02320802721

Shivansh Sharma, 36220802721

Certificate by Supervisor

This is to certify that Report titled “NewsTrueBytes” is submitted by **Khushi Kulshreshtha, Vibhuti Gupta, Priyanshu Gautam, Shivansh Sharma** in partial fulfillment of the requirement for the award of degree B.Tech in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. It is a record of the candidates own work carried out by them under my supervision. The matter embodied in this Report is original and has not been submitted for the award of any other degree.

25 October 2024 **Dr. Shweta Taneja**

Certificate by HOD

This is to certify that Report titled “NewsTrueBytes” is submitted by **Khushi Kulshreshtha, Vibhuti Gupta, Priyanshu Gautam, Shivansh Sharma** under the guidance of Dr Shweta Taneja in partial fulfillment of the requirement for the award of degree B.Tech in Computer Science & Engineering to BPIT, GGSIP University, Dwarka, Delhi. The matter embodied in this Report is original and has been duly approved for submission.

25 October 2024 **Prof. Achal Kaushik**

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Shweta Taneja for her invaluable guidance and support throughout the development of our NewsTrueBytes project. Her expertise and insights were crucial in helping us navigate the complexities of the three approaches we employed: Ensemble approach with BERT, Ensemble approach with DistilBERT, and CNN with BiLSTM.

Her continuous encouragement and advice pushed us to delve deeper into these advanced techniques and apply them effectively to our project. We are incredibly thankful for her time, patience, and the opportunity to work under her mentorship. This project has been a rewarding learning experience, and we are immensely grateful for her contributions to our success.

Khushi Kulshreshtha
Vibhuti Gupta
Priyanshu Gautam
Shivansh Sharma

Abstract

This report presents a comprehensive analysis of fake news detection using a two-stage approach. In the first stage, classification of fake and true news is performed using two techniques. The first technique applies deep learning models, specifically CNN + BiLSTM, achieving an accuracy of 98.7%. The second technique leverages a combination of deep learning and machine learning ensemble methods, including BERT, SVM, Random Forest, and Stacking, achieving an accuracy of 98.26%. In the second half of the project, a generative AI model, T5, is applied for news summarization. The T5 model is explored in depth, as it generates concise and coherent summaries, enhancing the usability of news articles.

Chapter 1

Introduction

In the modern digital landscape, the rapid spread of misinformation, particularly in the form of fake news, has become a major concern [1]. With the internet serving as the primary source of news for billions of people globally, distinguishing between real and fake news has never been more important. Fake news can have serious consequences, including influencing elections, damaging reputations, spreading panic, and undermining trust in institutions [5][6]. As social media platforms amplify the reach of news articles, the proliferation of fake news has led to a surge in demand for automated detection systems that can quickly and accurately classify articles as either real or fake [6][7].

This project seeks to address the fake news problem using advanced machine learning and deep learning techniques. Our approach involves two major components. First, we classify news articles as fake or true using two different methods. The first method employs a combination of Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks [12][8]. This model leverages both spatial features from the text (captured by CNN) and sequential dependencies (captured by BiLSTM). With an accuracy of 98.7%, this model demonstrates excellent performance in identifying fake news [12]. The second classification method uses BERT (Bidirectional Encoder Representations from Transformers) in combination with traditional machine learning algorithms such as Support Vector Machines (SVM), Random Forest, and Stacking ensembles. This combination also achieves a high accuracy of 98.26%, proving the robustness of the BERT model in understanding context and generating embeddings that help differentiate between fake and true news [16][15].

The second half of the project delves into the application of the T5 (Text-To-Text Transfer Transformer) model for news summarization. Given the vast amount of information available, summarizing news articles can significantly enhance user experience by allowing individuals to quickly grasp the essence of an article [8].

T5 is particularly suited for this task as itis apre-trained transformer model that excels in generating concise, high-quality summaries. By applying T5, we aim to create summaries that not only retain the most critical

information but also reduce the cognitive load on the reader, making it easier to process large volumes of news.

Overall, this project combines classification and summarization, providing a dual solution to combat fake news while enhancing content accessibility. This system will contribute towards maintaining the integrity of information in digital spaces and promoting a more informed public.

Chapter 2

Related Work

Fake news detection has been a focus of significant research, especially with the advent of social media platforms that allow the rapid dissemination of information. Early approaches to fake news detection primarily relied on manual fact-checking processes, which, although accurate, are time-consuming and infeasible given the scale of the problem. In response, machine learning techniques have gained prominence, offering faster and more scalable solutions to the fake news epidemic [5][7].

Several machine learning models have been used to classify news articles. Traditional models such as Naïve Bayes, Decision Trees, and SVM have been widely applied to detect fake news based on various features such as text content, author credibility, and social engagement metrics [8]. These models, while useful, often struggle to capture the complexities of language, especially with regard to context, tone, and nuance, which are critical when determining the veracity of a news article [10].

Deep learning models, particularly those involving neural networks, have recently shown remarkable success in natural language processing (NLP) tasks, including fake news detection [15]. CNNs have been employed to extract spatial patterns from text data, while RNNs and their variants (like LSTM and BiLSTM) have proven effective in capturing sequential dependencies in text [12]. Studies indicate that combining CNN and LSTM can yield robust models for text classification tasks, including fake news detection [13].

Transformers, especially BERT, have revolutionized NLP due to their ability to model deep contextual relationships within text. BERT, in particular, uses a bidirectional mechanism to understand the context of words from both directions, leading to superior performance in tasks such as text classification, question answering, and sentiment analysis [16]. Its application in fake news detection, often paired with ensemble methods like SVM or Random Forest, has shown significant promise in improving accuracy [11].

In the area of text summarization, early techniques relied on extractive methods, which involved selecting the most important sentences from a document. However, these methods often resulted in disjointed summaries. The emergence of generative models, particularly those based on transformers, has shifted the focus to abstractive summarization, where the model generates new text that conveys the essence of the original document [8]. Models like T5, trained on large datasets, have shown outstanding capabilities in producing coherent, readable summaries that capture the most important aspects of the source text [17].

This project builds on these advancements by employing a combination of CNN + BiLSTM for fake news detection, alongside BERT and ensemble methods for additional classification, and applying T5 for summarization. The hybrid approach ensures both high accuracy in classification and efficient content summarization, addressing both the detection and user engagement aspects of fake news [12][16].

Chapter 3

System Analysis and Design

3.1 : Software Requirement Specifications

- Hardware Requirements:
    - Processor: Intel Core i5 or higher
    - RAM: 8 GB or more
    - Storage: 100 GB available space
- Software Requirements:
    - Operating System: Windows 10 or higher
    - Programming Language: Python
    - Frameworks: TensorFlow, PyTorch
    - Libraries: Keras, scikit-learn, transformers (Hugging Face), NLTK

3.2 : Use Case Diagram

The system has two primary users: data scientists and journalists. The data scientist trains the classification models (CNN + BiLSTM and BERT + Ensemble), while the journalist uses the system to input news articles and obtain a classification (fake/true) and summarized version.

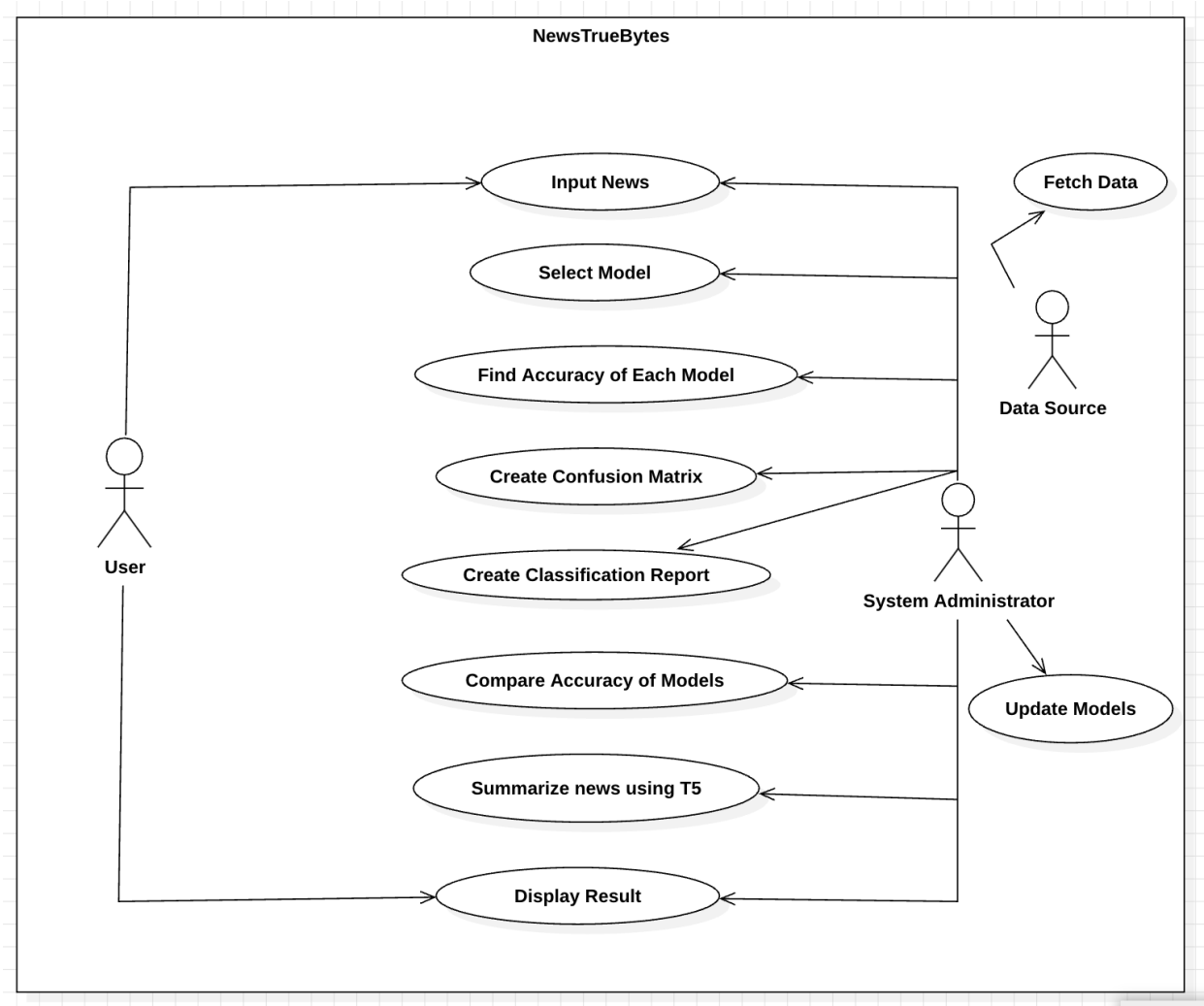

Fig 1: Use Case Diagram

3.3: Flow Chart / Activity Diagram

The system first processes the news dataset, then performs classification using CNN + BiLSTM or BERT + Ensemble models. Once classified, the T5 model generates a summary. The output includes both the classification result and a concise summary for each article.

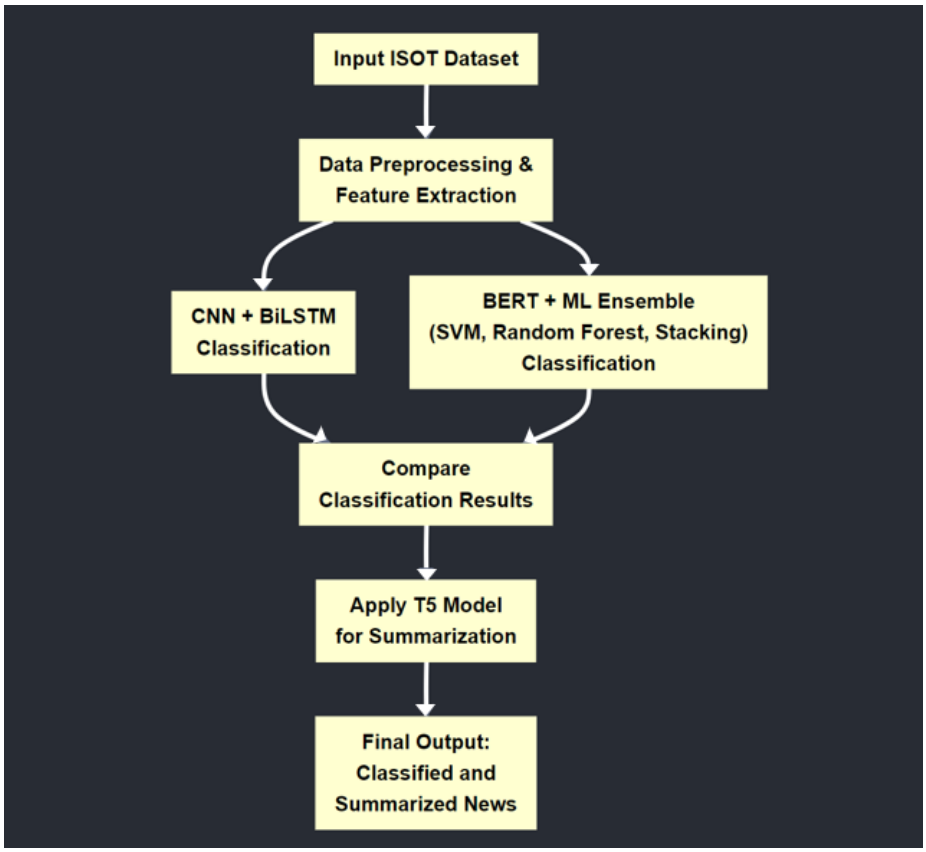

Fig 2: Flowchart

Chapter 4

Proposed Work

The proposed work for this project is divided into two main phases: classification and summarization. Both phases leverage advanced deep learning techniques to ensure high accuracy and performance. Below is a detailed explanation of the proposed methodology.

4.1 News Classification

In the first phase, we aim to classify news articles as either fake or true using multiple approaches that balance deep learning and traditional machine learning models for optimal performance. This multi-faceted approach allows us to compare different techniques and select models that perform best in handling nuanced language cues in news articles.

Model 1: CNN + BiLSTM Hybrid Model

The first approach employs a hybrid deep learning model combining Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory networks (BiLSTM). Typically used for image data, CNNs have proven effective in text classification due to their ability to capture spatial features, such as n-grams, which represent word sequences. In this model, CNN layers extract key patterns from the text, functioning like filters to identify significant n-grams. These features are then passed to the BiLSTM network, which processes them to capture the sequential relationships and dependencies between words in the text. As a bidirectional model, BiLSTM captures context from both directions of a sentence, making it particularly suited to understanding complex sentence structures and contextual nuances. This model combination enables the system to identify patterns in text effectively, achieving an accuracy rate of 98.7%.

Model 2: BERT + Ensemble Models

The second classification approach leverages BERT (Bidirectional Encoder Representations from Transformers), a pre-trained transformer model known for capturing deep semantic meaning in text. BERT's capability of transforming input text into embeddings encoding rich contextual information allows it to handle subtle variations in language, making it highly suitable for text classification tasks. The BERT-based embeddings are then fed into an ensemble of machine learning models, including Support Vector Machines (SVM), Random Forest, and Stacking. The ensemble technique enhances the robustness of predictions by combining the strengths of multiple models, yielding more accurate and reliable results. This BERT-ensemble approach achieves an accuracy rate of 98.26%, proving effective in classifying news articles even in cases of ambiguous or misleading content.

Model 3: DistilBERT + Ensemble Models

To further optimize computational efficiency and accuracy, we introduce an additional approach involving DistilBERT in combination with ensemble methods. DistilBERT is a distilled, lighter version of BERT that retains much of its contextual understanding while being more resource-efficient. This makes it a suitable choice for large-scale datasets or applications requiring quicker response times. DistilBERT generates embeddings with contextual information similar to BERT but with faster processing. These embeddings are fed into an ensemble of models like SVM, Random Forest, and Stacking. This setup achieves a balance between computational efficiency and classification accuracy. By leveraging DistilBERT's streamlined processing power, this model aims to achieve competitive accuracy rates while being more accessible for deployment on devices with limited processing capabilities.

Together, these three approaches offer a comprehensive strategy for accurately classifying news articles, catering to both high-precision requirements and computational efficiency, depending on the application's needs

4.2 News Summarization Using T5

The second phase of the project focuses on news summarization. Given the increasing volume of news articles published daily, summarization is critical for users who need to quickly grasp the content of a news article without reading the entire piece. The T5 model is a state-of-the-art transformer model trained in a text-to-text framework. It is pre-trained on a massive corpus, making it well-suited for tasks like translation, summarization, and question answering.

For this project, we fine-tune the T5 model on a custom news dataset to generate concise and coherent summaries of news articles. The T5 model is especially advantageous for abstractive summarization, where the summary consists of newly generated text that conveys the key points of the original article. Unlike extractive methods, which simply select and stitch together existing sentences, the T5 model generates human-like summaries that are easier to read and understand.

The proposed workflow ensures that after classifying an article as fake or true, the user can immediately obtain a summary of the article. This dual functionality not only helps users identify misinformation but also enhances the overall news consumption experience by providing quick, digestible content.

4.3 Word Cloud of True News

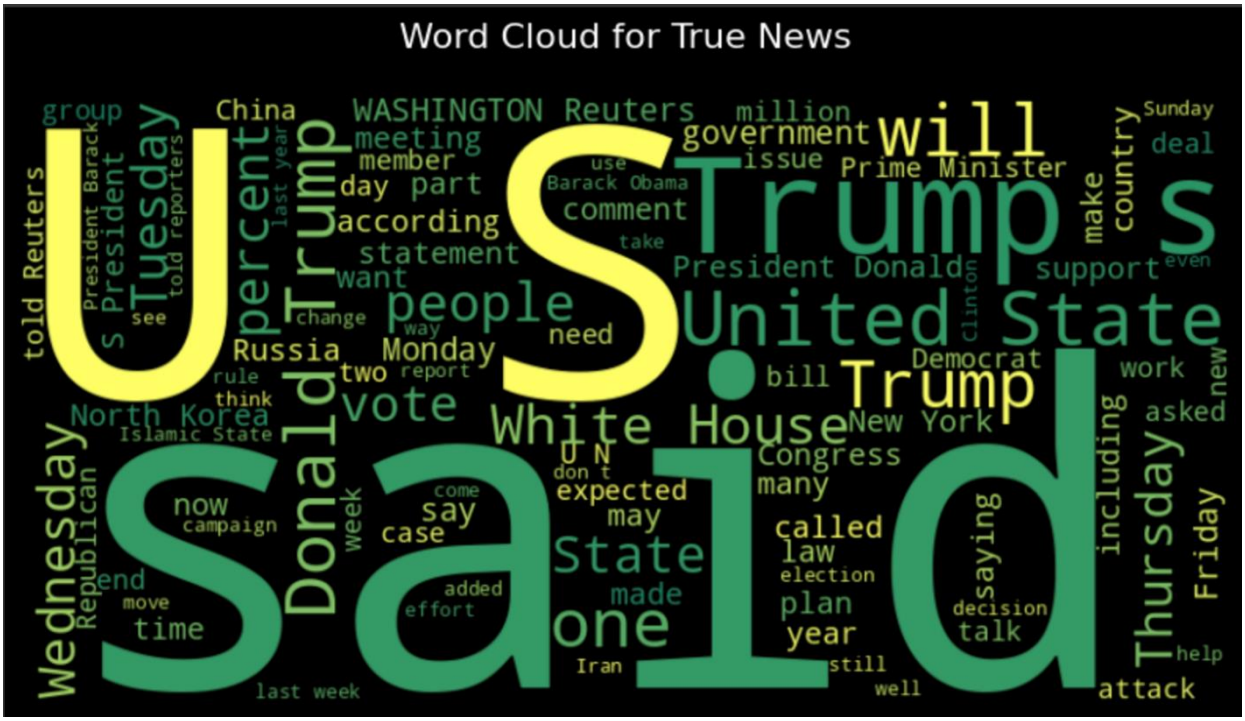

Fig 3: Word Cloud of True News

4.4Word Cloud of Fake News

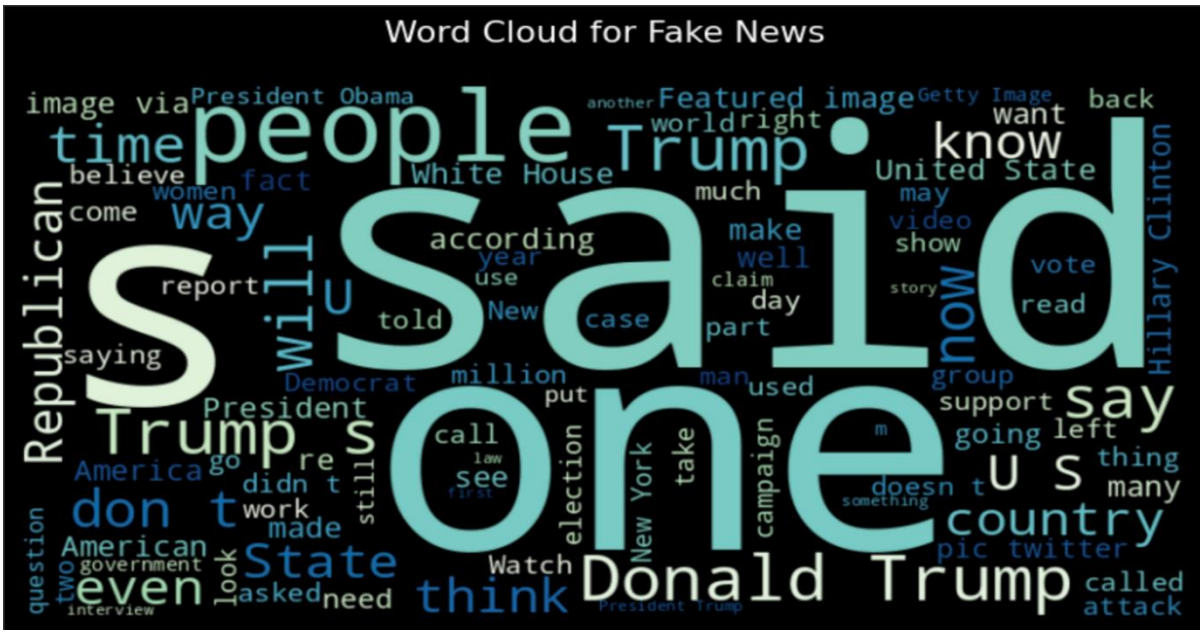

Fig 4: Word Cloud of Fake News

Chapter 5

Implementation and Results

5.1 Implementation of Classification Models

The ISOT dataset, which contains both fake and real news articles, was used for training and evaluating the models. The CNN + BiLSTM model and the DistilBERT + Ensemble model were trained using TensorFlow, achieving an accuracy of 98.7% on the test set. Additionally, BERT was fine-tuned using the Hugging Face library and combined with SVM, Random Forest, and Stacking ensemble techniques, resulting in a high accuracy of 98.26%. These approaches demonstrate effective classification performance, with the CNN + BiLSTM and DistilBERT + Ensemble models achieving slightly higher accuracy in comparison to the Bert model implementation.

5.2 Implementation of the T5 Model

For summarization, the T5 model was implemented using the transformers library. The model was fine-tuned on a custom news dataset to generate concise and relevant summaries. Initial results show the T5 model generated coherent summaries, reducing article length while retaining key information.

5.3 Results Comparison

The CNN + BiLSTM and DistilBERT + Ensemble model showed superior performance in terms of accuracy (98.7%) compared to the BERT + Ensemble approach (98.26%). However, the latter provided better generalization due to the ensemble nature of the model. The T5 model's summarizations were evaluated qualitatively, with most summaries being informative and relevant.

5.3.1 Accuracy of CNN + BiLSTM

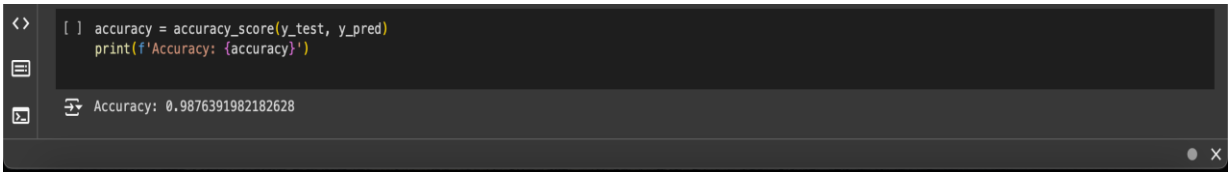

Fig 5: Accuracy of CNN & BILSTM

5.3.2 Confusion Matrix of CNN+BiLSTM

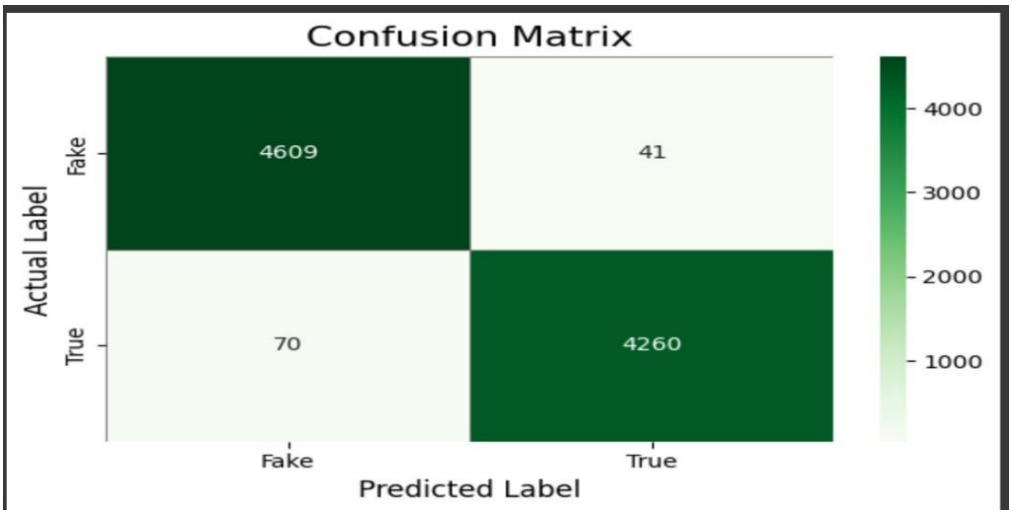

Fig 6: Confusion Matrix of CNN + BILSTM

5.3.3 ROC Curve of CNN+BILSTM

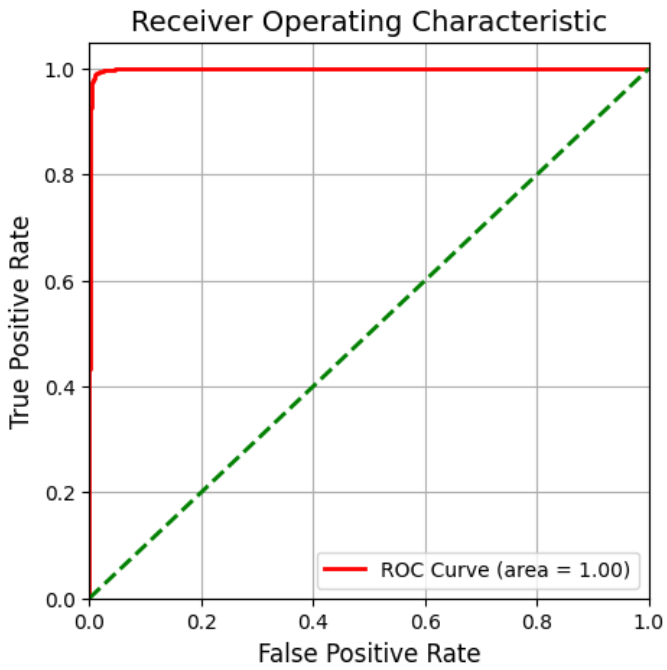

Fig 7: ROC Curve of CNN +BILSTM

5.3.4 Accuracy of Ensemble (SVM + RandomForest + Stacking) + BERT Model

```
[7]  from sklearn.svm import SVC
     from sklearn.ensemble import RandomForestClassifier, StackingClassifier

     # Define the base classifiers
     svm_clf = SVC(probability=True)  # SVM classifier
     rf_clf = RandomForestClassifier(n_estimators=100)  # Random Forest classifier

     # Define the stacking ensemble with SVM and Random Forest
     stacking_clf = StackingClassifier(
         estimators=[
             ('svm', svm_clf),
             ('rf', rf_clf)
         ],
         final_estimator=SVC(probability=True)
     )

     # Train the ensemble model
     stacking_clf.fit(X_train_bert.cpu().numpy(), y_train)  # Move tensor to CPU before converting to NumPy

     # Evaluate the ensemble model
     accuracy = stacking_clf.score(X_test_bert.cpu().numpy(), y_test)  # Move test tensor to CPU before converting to NumPy
     print(f'Ensemble Model Accuracy: {accuracy}')

Ensemble Model Accuracy: 0.9826280623608018
```

Fig 8: Accuracy of Ensemble + BERT

5.3.5 Confusion Matrix of ENSEMBLE + BERT

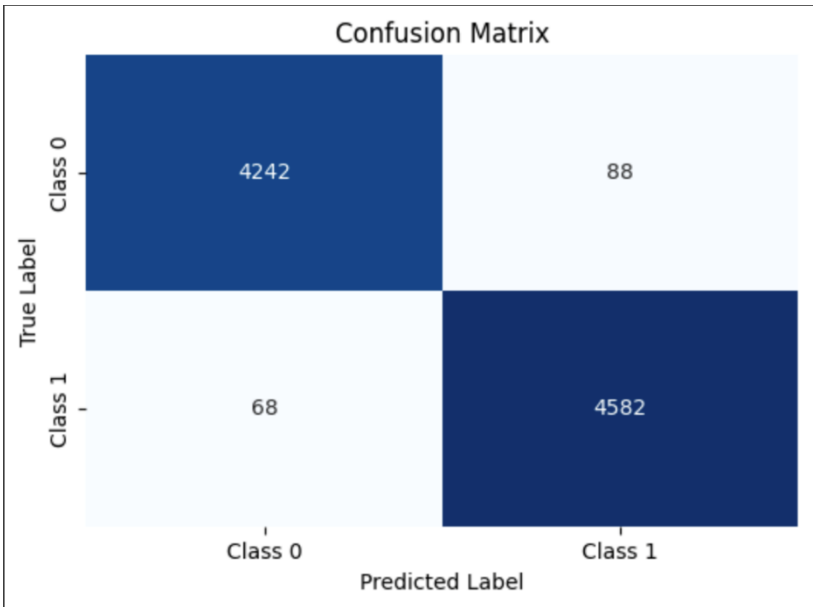

Fig 9: Confusion Matrix of Ensemble + BERT Model

5.3.6 ROC Curve Of ENSEMBLE+BERT

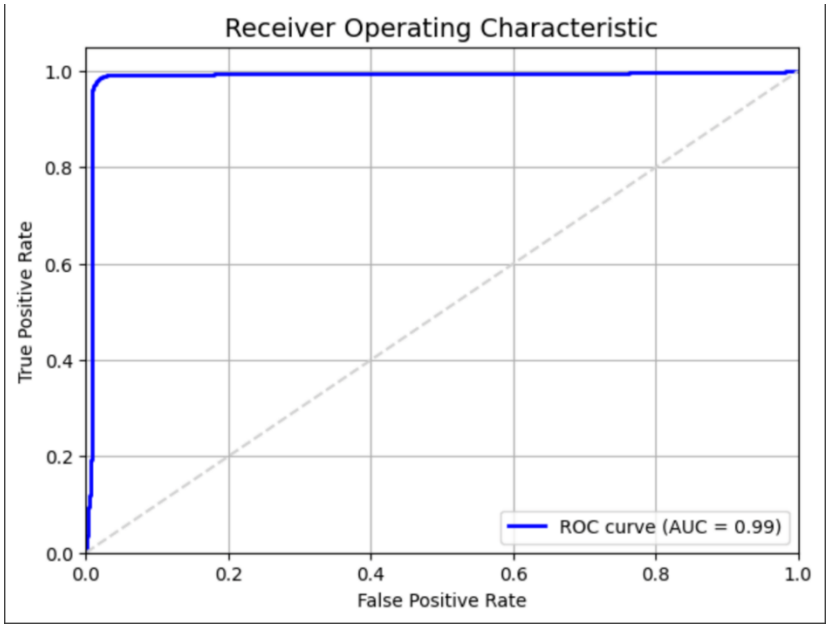

Fig 10: ROC Curve of Ensemble + BERT

5.3.7 Accuracy of Ensemble (SVM + RandomForest + Stacking) + DistilBERT

```
# Train the ensemble model on the entire training set
stacking_clf.fit(X_train_bert_scaled, y_train)

accuracy = stacking_clf.score(X_test_bert_scaled, y_test)
print(f'Ensemble Model Accuracy: {accuracy}')

# Predict on the test set
y_pred = stacking_clf.predict(X_test_bert_scaled)
```

```
Ensemble Model Accuracy: 0.9870824053452116
```

Fig 11: Accuracy of DistilBERT and ensemble

5.3.8 Confusion Matrix of Ensemble + DistilBERT Model

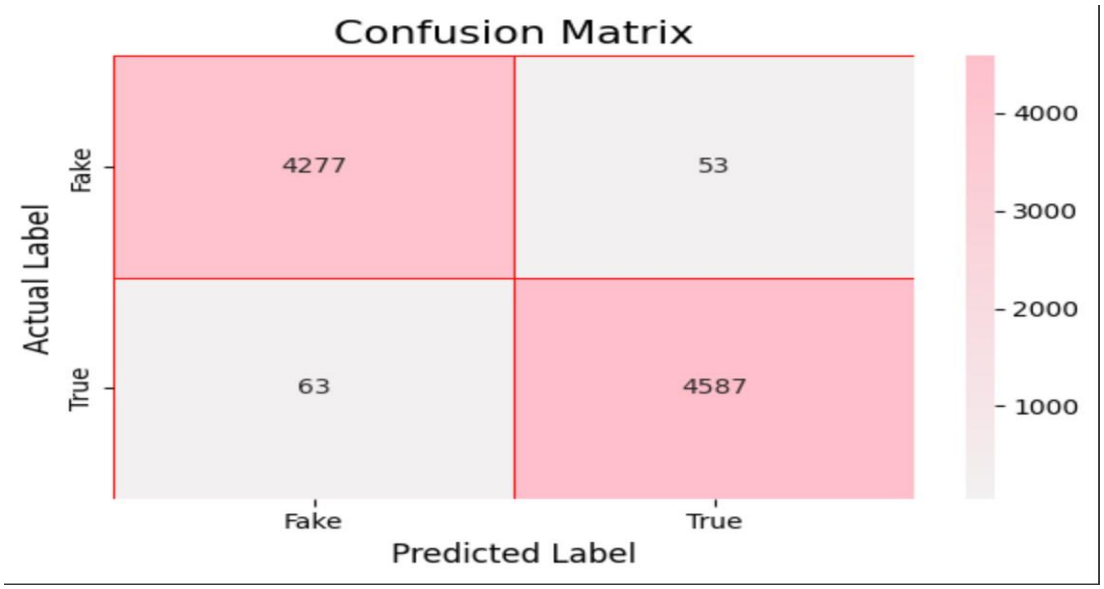

Fig 12: Confusion Matrix of Ensemble + DistilBERT Model

5.3.9 ROC Curve of ENSEMBLE + DISTILBERT

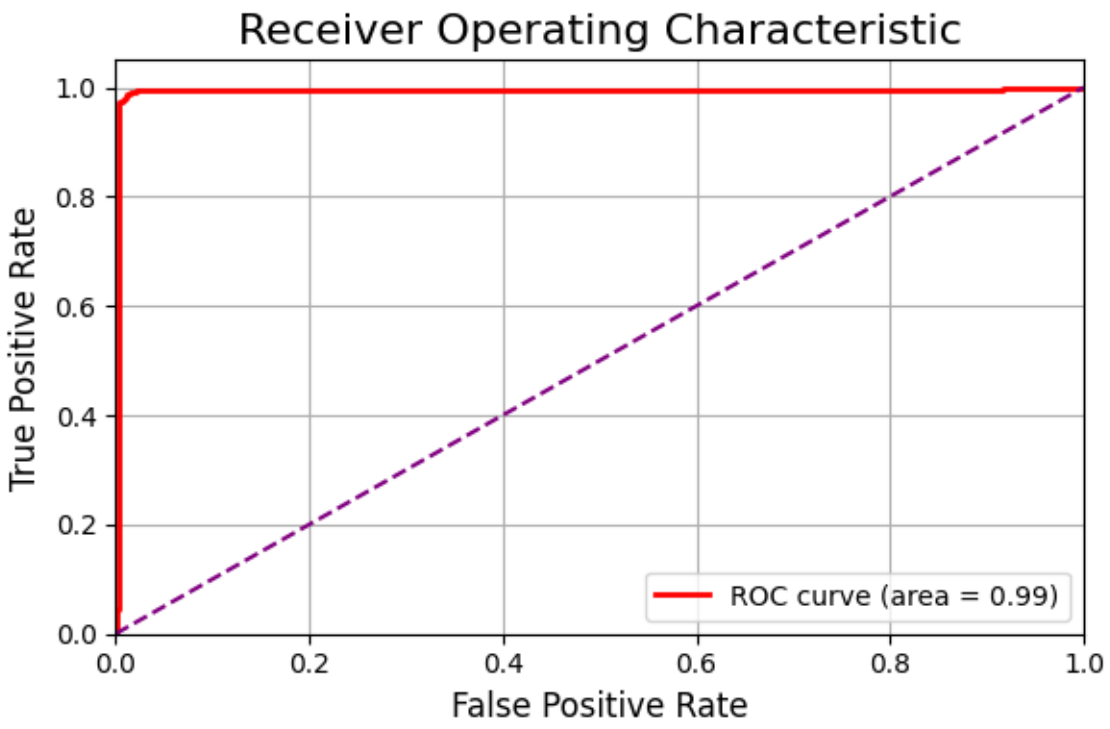

Fig 13: ROC Curve of Ensemble+ DistilBERT

5.3.10 Classification Report of All the Models

CODE	ACCURACY	PRECISION		RECALL		F1 SCORE	
		TRUE(0)	FALSE(1)	TRUE(0)	FALSE(1)	TRUE(0)	FALSE(1)
CNN+BILSTM	98.74	0.99	0.99	0.99	0.98	0.99	0.99
ENSEMBLE+DISTILBERT	98.7	0.99	0.99	0.99	0.99	0.99	0.99
ENSEMBLE+BERT	98.26	0.98	0.98	0.99	0.98	0.98	0.98

Table 1: Classification Report of All Models

5.3.11 Summarization Using T5

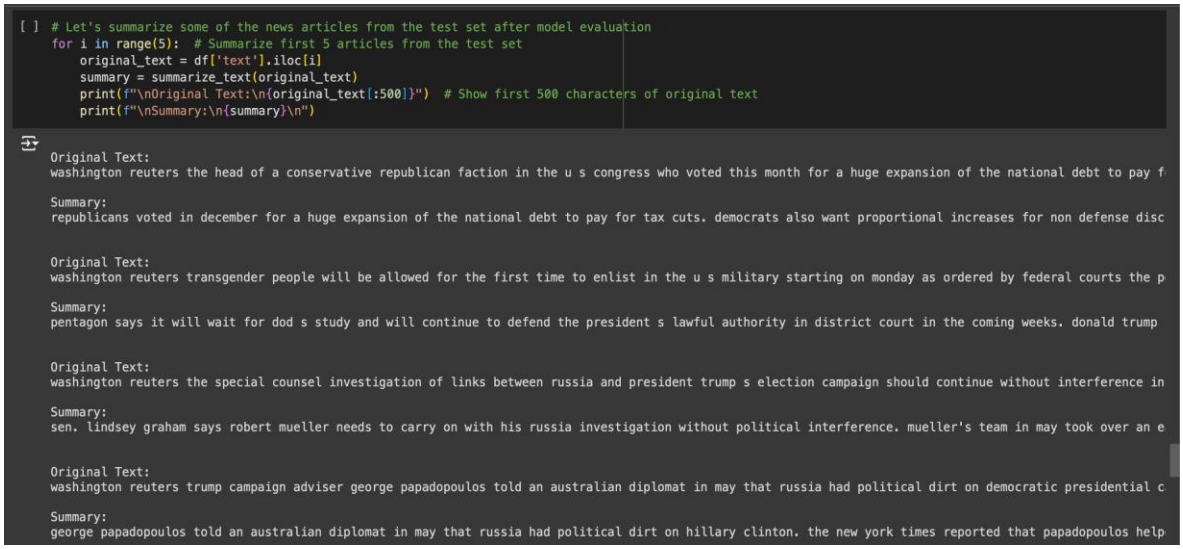

Fig 14: Summarization Using T5

Chapter 6

Conclusion

This project successfully demonstrates a two-stage approach to combating fake news. In the first stage, we implemented two methods for classifying news articles as fake or true. The CNN + BiLSTM model and DistilBert with ensemble approach achieved a high accuracy of 98.7%, showcasing the effectiveness of combining spatial and temporal features in text classification. Additionally, the BERT + Ensemble approach, which achieved an accuracy of 98.26%, proved to be a robust and reliable model, particularly in understanding the context and nuances of the text.

The second phase of the project applied the T5 model for news summarization, generating concise, readable summaries of news articles. The summarization feature enhances the usability of the system, allowing users to quickly digest news content while avoiding fake news. The T5 model's ability to generate abstractive summaries ensures that the output is not just a collection of selected sentences but a coherent, meaningful synthesis of the article.

In conclusion, this project addresses both the detection of fake news and the need for efficient news consumption. The hybrid approach to classification, combined with the generative power of the T5 model, makes this system a comprehensive solution for both identifying and summarizing news articles. This project represents a step forward in the fight against misinformation in the digital age.

Chapter 7

Future Work

While this project has demonstrated the potential of combining deep learning and generative AI for fake news detection and summarization, there are several avenues for future work. One of the most promising directions is to fine-tune the classification models on larger, more diverse datasets. This would improve the generalization capabilities of the models, making them more robust against new forms of misinformation and better suited for deployment in varied cultural and linguistic contexts. Additionally, applying techniques like active learning, where the model continuously improves by learning from new data, could further enhance its accuracy and adaptability in identifying evolving fake news patterns.

In terms of summarization, exploring newer generative models such as GPT-4 or further refining the T5 model could lead to even more coherent, contextually-aware summaries. Reinforcement learning could also be employed to improve the summarization process by rewarding the model for generating summaries that are not only informative but also concise and easily comprehensible for end-users. Moreover, experimentation with abstractive summarization techniques tailored to news context could add a layer of nuance to the summaries, making them closer to human comprehension.

Another critical area for future work is the integration of explainability tools. As AI systems increasingly play a crucial role in information verification, it is essential to provide transparent and explainable decisions. Techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) could be incorporated to help users understand the rationale behind classifying an article as fake or true. Such transparency would enhance trust in the system and promote its adoption among journalists, researchers, and readers.

Moreover, integrating the system with real-time data from social media platforms or major news outlets could make it a powerful, on-demand tool for journalists, researchers, and everyday users seeking to quickly assess the credibility of news articles. The application of transfer learning, where models trained on one domain are adapted to new, related domains, could also be explored to make the system adaptable to other types of misinformation, including deepfakes, manipulated images, or misleading multimedia content.

Finally, collaboration with fact-checking organizations could lead to a more comprehensive and reliable system, where AI-based predictions are verified by human fact-checkers. This hybrid approach would ensure that the system not only detects fake news but also provides thoroughly vetted, trustworthy information, paving the way for a stronger, AI-supported information verification ecosystem that is both effective and publicly trusted.

REFERENCES

[1] A. Abdulrahman and M. Baykara, “Fake News Detection Using Machine Learning and Deep Learning Algorithms,” *2020 International Conference on Advanced Science and Engineering (ICOASE)*, pp. 18–23, Dec. 2020, doi: 10.1109/icoase51841.2020.9436605.

[2] A. A. A. A. E. Al, “Detecting Fake News using Machine Learning: A Systematic Literature Review,” *Psychology and Education Journal*, vol. 58, no. 1, pp. 1932–1939, Jan. 2021, doi: 10.17762/pae.v58i1.1046.

[3] M. Devi, R. Priyanka, P. Surendra, B. Priyanka, and Ch. Nikhila, "Fake news detection using machine learning," SSRN Electronic Journal, vol. 6, pp. 104-109, 2019.

[4] P. Gupta, S. Gandhi, and B. R. Chakravarthi, “Leveraging Transfer learning techniques-BERT, RoBERTa, ALBERT and DistilBERT for Fake Review Detection,” *Forum for Information Retrieval Evaluation*, vol. 29, pp. 75–82, Dec. 2021, doi: 10.1145/3503162.3503169.

[5] M. F. Mridha, A. Jannat Keya, Md. A. Hamid, M. M. Monowar, and Md. S. Rahman, “A Comprehensive Review on Fake News Detection With Deep Learning,” *IEEE Xplore*, Nov. 18, 2021. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9620068

[6] B. Suri, S. Taneja, S. Aggarwal, and V. R. Sharma, “Fake news detection tool (FNDT): Shield against sentimental deception,” *Journal of Information and Optimization Sciences*, vol. 41, no. 6, pp. 1513–1524, doi: 10.1080/02522667.2020.1802125.

[7] H. F. Villela, F. Corrêa, J. S. De Araújo Nery Ribeiro, A. Rabelo, and D. B. F. Carvalho, “Fake news detection: a systematic literature review of machine learning algorithms and datasets,” *Journal on Interactive Systems*, vol. 14, no. 1, pp. 47–58, Mar. 2023, doi: 10.5753/jis.2023.3020.

[8] I. K. Sastrawan, I. P. A. Bayupati, and D. M. S. Arsa, “Detection of fake news using deep learning CNN–RNN based methods,” *ICT Express*, vol. 8, no. 3, pp. 396–408, Oct. 2021, doi: 10.1016/j.icte.2021.10.003.

[9] Z. Khanam, B. N. Alwasel, H. Sirafi, and M. Rashid, “Fake news detection using machine learning approaches,” *IOP Conference Series Materials Science and Engineering*, vol. 1099, no. 1, p. 012040, Mar. 2021, doi: 10.1088/1757-899x/1099/1/012040.

[10] S. Murugesan and K. Pachamuthu, “Fake News Detection in the Medical Field Using Machine Learning Techniques,” *International Journal of Safety and Security Engineering*, vol. 12, no. 6, pp. 723–727, doi: 10.18280/ijsse.120608.

[11] J. Alghamdi, Y. Lin, and S. Luo, "Modeling Fake News Detection Using BERT-CNN-BiLSTM Architecture," *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 354–357, Aug. 2022, doi: 10.1109/mipr54900.2022.00069.

[12] N. Rani, P. Das, and A. K. Bhardwaj, "A hybrid deep learning model based on CNN-BiLSTM for rumor detection," *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, pp. 1423–1427, Jul. 2021, doi: 10.1109/icces51350.2021.9489214.

[13] A. K. Yadav *et al.*, "Fake news detection using hybrid deep learning method," *SN Computer Science*, vol. 4, no. 6, Nov. 2023, doi: 10.1007/s42979-023-02296-w.

[14] Y. Fang, J. Gao, C. Huang, H. Peng, and R. Wu, "Self Multi-Head Attention-based Convolutional Neural Networks for fake news detection," *PLoS ONE*, vol. 14, no. 9, p. e0222713, Sep. 2019, doi: 10.1371/journal.pone.0222713.

[15] R. K. Kaliyar, A. Goswami, and P. Narang, "Multiclass Fake News Detection using Ensemble Machine Learning," *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, pp. 103–107, Dec. 2019, doi: 10.1109/iacc48062.2019.8971579.

[16] R. K. Kaliyar, A. Goswami, and P. Narang, "FakeBERT: Fake news detection in social media with a BERT-based deep learning approach," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 11765–11788, Jan. 2021, doi: 10.1007/s11042-020-10183-2.

[17] P. Bahad, P. Saxena, and R. Kamal, "Fake News Detection using Bi-directional LSTM-Recurrent Neural Network," *Procedia Computer Science*, vol. 165, pp. 74–82, Jan. 2019, doi: 10.1016/j.procs.2020.01.072.

[18] I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, "Fake news detection using machine learning ensemble methods," *Complexity*, vol. 2020, pp. 1–11, Oct. 2020, doi: 10.1155/2020/8885861.

[19] S. Elyassami, S. Alseiari, M. ALZaabi, A. Hashem, and N. Aljahoori, "Fake news detection using ensemble learning and machine learning algorithms," in *Studies in computational intelligence*, 2021, pp. 149–162. doi: 10.1007/978-3-030-90087-8_7.

[20] M. A. Pradhan, "Fake News Detection Methods: Machine Learning approach," International Journal for Research in Applied Science and Engineering Technology, vol. 8, no. 7, pp. 971–975, Jul. 2020, doi: 10.22214/ijraset.2020.29630.

[21] A. M. Ali, F. A. Ghaleb, B. A. S. Al-Rimy, F. J. Alsolami, and A. I. Khan, "Deep Ensemble Fake news detection model using sequential deep learning technique," *Sensors*, vol. 22, no. 18, p. 6970, Sep. 2022, doi: 10.3390/s22186970.

[22] S. Murugesan and K. Pachamuthu, "Fake News Detection in the Medical Field Using Machine Learning Techniques," *International Journal of Safety and Security Engineering*, vol. 12, no. 6, pp. 723–727, doi: 10.18280/ijsse.120608.

CODING

1. CNN+BiLSTM+T5

```
# Import necessary libraries
import pandas as pd

import numpy as np

import re

import nltk
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# The Tokenizer is now located in tensorflow.keras.utils in Keras 3 and
above
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Import other keras modules from tensorflow.keras instead of just keras
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D, LSTM,
Bidirectional, Dense, Dropout
from tensorflow.keras.optimizers import Adam
nltk.download('stopwords')
from nltk.corpus import stopwords
# Load the dataset
true_news = pd.read_csv('/content/True.csv')
fake_news = pd.read_csv('/content/Fake.csv')
# Add a label for both datasets
true_news['label'] = 1
fake_news['label'] = 0
# Combine the datasets
df = pd.concat([true_news, fake_news], axis=0).reset_index(drop=True)
# Text cleaning function
def clean_text(text):
text = text.lower()
text = re.sub(r'\[.*?\]', '', text)
text = re.sub(r'http\S+', '', text)
text = re.sub(r'<.*?>+', '', text)
text = re.sub(r'[^a-zA-Z]', ' ', text)
text = re.sub(r'\s+', ' ', text).strip()
return text

# Clean the text in the dataset

df['text'] = df['text'].apply(clean_text)
# Tokenization and padding
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(df['text'])
vocab_size = len(tokenizer.word_index) + 1
# Convert the text to sequences

sequences = tokenizer.texts_to_sequences(df['text'])
# Pad the sequences to ensure uniform input length
max_len = 300

X = pad_sequences(sequences, maxlen=max_len)
# Labels
```

```
y = df['label'].values
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# CNN + BiLSTM Model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=100,
input_length=max_len))
model.add(Conv1D(filters=128, kernel_size=5, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Bidirectional(LSTM(100, return_sequences=False)))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
# Compile the model
model.compile(loss='binary_crossentropy',
optimizer=Adam(learning_rate=1e-4), metrics=['accuracy'])
# Train the model
history = model.fit(X_train, y_train, epochs=5, batch_size=64,
validation_split=0.1, verbose=1)
# Evaluate the model
y_pred = model.predict(X_test)
y_pred = np.round(y_pred).astype(int)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

def plot_classification_report(y_test, y_pred):
    # Generate the classification report as a dictionary
    report = classification_report(y_test, y_pred, output_dict=True)

    # Convert the report dictionary to a Pandas DataFrame for easier
manipulation
    report_df = pd.DataFrame(report).transpose()

    # Select the desired rows to display ('0', '1', 'accuracy', 'macro
avg', 'weighted avg')
    report_df = report_df.loc[['0', '1', 'accuracy', 'macro avg',
'weighted avg'], ['precision', 'recall', 'f1-score', 'support']]

#Create a heatmap to visualize the classification report with a yellow
color palette
    plt.figure(figsize=(8, 6))
    sns.heatmap(report_df.iloc[:, :-1], annot=True, cmap='YlGn',
cbar=False, fmt='.2f', linewidths=0.5)

    plt.title('Classification Report', fontsize=16)
    plt.ylabel('Classes', fontsize=12)
    plt.xlabel('Metrics', fontsize=12)

    # Adjust layout for better display
    plt.tight_layout()
    plt.show()
plot_classification_report(y_test, y_pred)
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(y_test, y_pred):


    # Set up the figure and axes
    plt.figure(figsize=(6, 4))

    # Create a heatmap for the confusion matrix
    sns.heatmap(cm, annot=True, fmt='d', cmap='Greens',
                xticklabels=['Fake', 'True'], yticklabels=['Fake',
'True'],
                linewidths=0.5, linecolor='gray')

    # Customize the title and labels
    plt.title('Confusion Matrix', fontsize=16)
    plt.ylabel('Actual Label', fontsize=12)
    plt.xlabel('Predicted Label', fontsize=12)

    # Adjust layout for better spacing
    plt.tight_layout()
    plt.show()
plot_confusion_matrix(y_test, y_pred)
# Import necessary libraries for T5
from transformers import T5Tokenizer, T5ForConditionalGeneration
import torch
# Load pre-trained T5 tokenizer and model for text summarization
tokenizer = T5Tokenizer.from_pretrained('t5-small')
model_t5 = T5ForConditionalGeneration.from_pretrained('t5-small')
# Function for generating summaries using T5
def summarize_text(text, max_length=150, min_length=40):
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt",
max_length=512, truncation=True)
    summary_ids = model_t5.generate(inputs, max_length=max_length,
min_length=min_length, length_penalty=2.0, num_beams=4,
early_stopping=True)
    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
    return summary
# Let's summarize some of the news articles from the test set after model
evaluation
for i in range(5):  # Summarize first 5 articles from the test set
    original_text = df['text'].iloc[i]
    summary = summarize_text(original_text)
    print(f"\nOriginal Text:\n{original_text[:500]}")  # Show first 500
characters of original text
    print(f"\nSummary:\n{summary}\n")
from transformers import T5Tokenizer, T5ForConditionalGeneration
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer()
# Final Display: Predicting if the News is Fake or True
def predict_news(news_article):
    # Clean the input news article
    cleaned_article = clean_text(news_article)

    # Tokenize and pad the input text
    sequence = tokenizer.texts_to_sequences([cleaned_article])
    padded_sequence = pad_sequences(sequence, maxlen=max_len)
```

```
    # Predict the label (0 = Fake, 1 = True)
    prediction = model.predict(padded_sequence)
    prediction = np.round(prediction).astype(int)[0][0]  # Get the binary
label (0 or 1)

    # Display the result
    if prediction == 1:
        print("\nThe news is True.")
    else:
        print("\nThe news is Fake.")
# Example usage of the prediction function
news_article = "Shelling kills 14 in Russia’s Belgorod following Moscow’s
aerial attacks across Ukraine"
predict_news(news_article)
```

2. Ensemble (SVM + RandomForest + Stacking) + BERT+T5

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
import re
import torch

# Load the datasets
real_news = pd.read_csv('/content/True.csv')
fake_news = pd.read_csv('/content/Fake.csv')

# Add labels
real_news['label'] = 0  # 0 for real news
fake_news['label'] = 1  # 1 for fake news

# Combine the datasets
news_data = pd.concat([real_news, fake_news],
axis=0).reset_index(drop=True)

# Preprocess the text
def clean_text(text):
    text = re.sub(r'\W', ' ', text)  # Remove special characters
    text = re.sub(r'\s+', ' ', text)  # Remove multiple spaces
    text = text.lower()  # Convert to lowercase
    return text

news_data['text'] = news_data['text'].apply(clean_text)

# Split into training and test sets
X = news_data['text']
y = news_data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Convert text to TF-IDF features
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train).toarray()
X_test_tfidf = tfidf_vectorizer.transform(X_test).toarray()
# Check if X_train contains only strings
X_train = [str(text) for text in X_train]
```

```
X_test = [str(text) for text in X_test]
from transformers import BertTokenizer, BertModel
import torch

# Load pre-trained BERT model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = BertModel.from_pretrained('bert-base-uncased')
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
bert_model.to(device)

# Function to extract BERT embeddings for a batch of texts
def get_bert_embeddings_batch(texts):
    # Tokenize all texts in the batch together
    inputs = tokenizer(texts, return_tensors='pt', max_length=512,
truncation=True, padding=True)

    # Move inputs to the same device as the BERT model (GPU/CPU)
    inputs = {key: value.to(bert_model.device) for key, value in
inputs.items()}

    with torch.no_grad():
        # Get the model outputs
        outputs = bert_model(**inputs)

    # Take the mean of the hidden states across the token dimension
(dim=1) to get embeddings
    embeddings = torch.mean(outputs.last_hidden_state, dim=1)

    return embeddings


def get_bert_embeddings_in_batches(texts, batch_size):
    embeddings = []
    for i in range(0, len(texts), batch_size):
        # Get batch of texts
        batch_texts = texts[i:i + batch_size]
        # Get embeddings for the batch
        batch_embeddings = get_bert_embeddings_batch(batch_texts)
        # Append the embeddings
        embeddings.append(batch_embeddings)
    return torch.cat(embeddings)
# Extract BERT embeddings for train and test sets
batch_size = 32
X_train_bert = get_bert_embeddings_in_batches(X_train, batch_size)
X_test_bert = get_bert_embeddings_in_batches(X_test, batch_size)
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, StackingClassifier

# Define the base classifiers
svm_clf = SVC(probability=True)   # SVM classifier
rf_clf = RandomForestClassifier(n_estimators=100)  # Random Forest
classifier

# Define the stacking ensemble with SVM and Random Forest
stacking_clf = StackingClassifier(
    estimators=[
        ('svm', svm_clf),
        ('rf', rf_clf)
```

```
    ],
    final_estimator=SVC(probability=True)
)

# Train the ensemble model
stacking_clf.fit(X_train_bert.cpu().numpy(), y_train)  # Move tensor to
CPU before converting to NumPy

# Evaluate the ensemble model
accuracy = stacking_clf.score(X_test_bert.cpu().numpy(), y_test)  # Move
test tensor to CPU before converting to NumPy
print(f'Ensemble Model Accuracy: {accuracy}')
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import classification_report, confusion_matrix

# Move the test data tensor to CPU and convert to NumPy
X_test_bert_cpu = X_test_bert.cpu().numpy()

# Predict on the test set
y_pred = stacking_clf.predict(X_test_bert_cpu)

# Generate the classification report and confusion matrix
class_report = classification_report(y_test, y_pred, output_dict=True)  #
Convert to dict for DataFrame
cm = confusion_matrix(y_test, y_pred)

# Plot Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0',
'Class 1'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# Plot Classification Report
class_report_df = pd.DataFrame(class_report).transpose()  # Convert
classification report to DataFrame
plt.figure(figsize=(10, 6))
sns.heatmap(class_report_df.iloc[:-1, :-1], annot=True, cmap='YlGnBu',
cbar=False, linewidths=0.5)
plt.title('Classification Report')
plt.show()
from transformers import T5Tokenizer, T5ForConditionalGeneration

# Load pre-trained T5 model and tokenizer for summarization
t5_model = T5ForConditionalGeneration.from_pretrained('t5-small')  # You
can also use 't5-base' or 't5-large'
t5_tokenizer = T5Tokenizer.from_pretrained('t5-small')

# Move the model to the appropriate device (GPU or CPU)
t5_model.to(device)
# Function to summarize a given text
def summarize_text(text, max_length=150):
    # Prepend "summarize: " to the input text, as T5 expects this prompt
```

```
    input_text = "summarize: " + text

    # Tokenize the input text
    inputs = t5_tokenizer.encode(input_text, return_tensors='pt',
max_length=512, truncation=True)
    inputs = inputs.to(device)  # Move inputs to device

    # Generate summary (you can adjust max_length and num_beams for better
summaries)
    summary_ids = t5_model.generate(inputs, max_length=max_length,
num_beams=4, length_penalty=2.0, early_stopping=True)

    # Decode the generated summary
    summary = t5_tokenizer.decode(summary_ids[0],
skip_special_tokens=True)
    return summary
def check_news(row):
    if row['label'] == 'fake':  # Assuming 'label' column contains 'fake'
or 'real'
        return "The news is fake"
    else:
        summary = summarize_text(row['text'])
        return f"The news is true. Summary: {summary}"

# Apply the function to each row
real_news['result'] = real_news.apply(check_news, axis=1)

# Display a few results
for i in range(4):  # Display first 4 rows with their results
    print(f"Original Text: {real_news['text'].iloc[i]}")
    print(f"Summary: {real_news['result'].iloc[i]}\n")
```

3. Ensemble (SVM + RandomForest + Stacking) + DistilBERT+T5

```
import pandas as pd
from sklearn.model_selection import train_test_split
import re
# Load the datasets
real_news = pd.read_csv('True.csv')
fake_news = pd.read_csv('Fake.csv')

# Add labels
real_news['label'] = 0  # 0 for real news
fake_news['label'] = 1  # 1 for fake news

# Combine the datasets
news_data = pd.concat([real_news, fake_news],
axis=0).reset_index(drop=True)
# Preprocess the text
def clean_text(text):
    text = re.sub(r'\W', ' ', text)  # Remove special characters
    text = re.sub(r'\s+', ' ', text)  # Remove multiple spaces
    text = text.lower()  # Convert to lowercase
    return text

news_data['text'] = news_data['text'].apply(clean_text)
# Split into training and test sets
X = news_data['text']
```

```
y = news_data['label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
from transformers import DistilBertTokenizer, DistilBertModel
import torch
from torch.utils.data import DataLoader, TensorDataset

# Load pre-trained DistilBERT model and tokenizer
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
distilbert_model = DistilBertModel.from_pretrained('distilbert-base-
uncased')

# Device configuration (use GPU if available)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
distilbert_model = distilbert_model.to(device)
# Tokenize and encode text into tensors for batch processing
def encode_texts(texts, tokenizer, max_len=512):
    encodings = tokenizer.batch_encode_plus(
        texts.tolist(),
        truncation=True,
        padding='max_length',
        max_length=max_len,
        return_tensors='pt'
    )
    return encodings['input_ids'], encodings['attention_mask']

# Encode training and test data
X_train_input_ids, X_train_attention_mask = encode_texts(X_train,
tokenizer)
X_test_input_ids, X_test_attention_mask = encode_texts(X_test, tokenizer)
# Create TensorDatasets and DataLoaders for training and test sets
train_dataset = TensorDataset(X_train_input_ids, X_train_attention_mask)
test_dataset = TensorDataset(X_test_input_ids, X_test_attention_mask)

# Define DataLoader with batch size
batch_size = 32  # Adjust batch size depending on your GPU/CPU memory
train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=batch_size,
shuffle=False)
# Function to extract DistilBERT embeddings in batches
def get_distilbert_embeddings(dataloader, model):
    model.eval()  # Set model to evaluation mode
    embeddings_list = []

    with torch.no_grad():  # Disable gradient calculation
        for batch in dataloader:
            # Unpack the batch
            input_ids, attention_mask = batch
            input_ids = input_ids.to(device)
            attention_mask = attention_mask.to(device)

            # Get DistilBERT output
            outputs = model(input_ids=input_ids,
attention_mask=attention_mask)
```

```
            # Get the mean of the hidden states for each sentence
(sentence embeddings)
            embeddings = torch.mean(outputs.last_hidden_state, dim=1)
            embeddings_list.append(embeddings.cpu())

    return torch.cat(embeddings_list, dim=0)
# Extract DistilBERT embeddings for training and test sets
X_train_bert = get_distilbert_embeddings(train_loader, distilbert_model)
X_test_bert = get_distilbert_embeddings(test_loader, distilbert_model)
from sklearn.svm import SVC  # Import SVC (Support Vector Classifier)
from sklearn.ensemble import RandomForestClassifier  # Import
RandomForestClassifier
from sklearn.ensemble import StackingClassifier  # Import
StackingClassifier
from sklearn.preprocessing import StandardScaler  # Import StandardScaler
for scaling
from sklearn.model_selection import cross_val_score  # Import
cross_val_score for cross-validation
from sklearn.metrics import classification_report, confusion_matrix  #
Metrics for evaluation


# Standardize the BERT embeddings before trainin
scaler = StandardScaler()

# Fit the scaler on training data and transform both train and test sets
X_train_bert_scaled = scaler.fit_transform(X_train_bert.numpy())
X_test_bert_scaled = scaler.transform(X_test_bert.numpy())
# Define the base classifiers
svm_clf = SVC(probability=True)   # SVM classifier
rf_clf = RandomForestClassifier(n_estimators=100)  # Random Forest
classifier

# Define the stacking ensemble with SVM and Random Forest
stacking_clf = StackingClassifier(
    estimators=[
        ('svm', svm_clf),
        ('rf', rf_clf)
    ],
    final_estimator=SVC(probability=True)
# Train the ensemble model on the entire training set
stacking_clf.fit(X_train_bert_scaled, y_train)

accuracy = stacking_clf.score(X_test_bert_scaled, y_test)
print(f'Ensemble Model Accuracy: {accuracy}')

# Predict on the test set
y_pred = stacking_clf.predict(X_test_bert_scaled)

# Classification report
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

def plot_classification_report(y_test, y_pred):
    # Generate the classification report as a dictionary
    report = classification_report(y_test, y_pred, output_dict=True)
```

```
    # Convert the report dictionary to a Pandas DataFrame for easier
manipulation
    report_df = pd.DataFrame(report).transpose()

    # Select the desired rows to display ('0', '1', 'accuracy', 'macro
avg', 'weighted avg')
    report_df = report_df.loc[['0', '1', 'accuracy', 'macro avg',
'weighted avg'], ['precision', 'recall', 'f1-score', 'support']]

    # Create a heatmap to visualize the classification report with a pink
color palette
    plt.figure(figsize=(8, 6))
    custom_palette = sns.light_palette("pink", as_cmap=True)
    sns.heatmap(report_df.iloc[:, :-1], annot=True, cmap=custom_palette,
cbar=False, fmt='.2f', linewidths=0.5)

    plt.title('Classification Report', fontsize=16)
    plt.ylabel('Classes', fontsize=12)
    plt.xlabel('Metrics', fontsize=12)

    # Adjust layout for better display
    plt.tight_layout()
    plt.show()
plot_classification_report(y_test, y_pred)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(y_test, y_pred):
    # Generate the confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Set up the figure and axes
    plt.figure(figsize=(6, 4))

    # Create a custom pink color palette
    custom_pink_palette = sns.light_palette("pink", as_cmap=True)

    # Create a heatmap for the confusion matrix with a custom pink color
palette
    sns.heatmap(cm, annot=True, fmt='d', cmap=custom_pink_palette,
                xticklabels=['Fake', 'True'], yticklabels=['Fake',
'True'],
                linewidths=0.5, linecolor='red')

    # Customize the title and labels
    plt.title('Confusion Matrix', fontsize=16)
    plt.ylabel('Actual Label', fontsize=12)
    plt.xlabel('Predicted Label', fontsize=12)

    # Adjust layout for better spacing
    plt.tight_layout()
    plt.show()

# Example usage
plot_confusion_matrix(y_test, y_pred)
```