HYBRID APPROACH FOR INTELLIGENT FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING

BY

EZE IFEANYI VALENTINE
GOU/U22/CSC/800

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, UGWUOMU NIKE, ENUGU, ENUGU STATE.

JUNE 2026

HYBRID APPROACH FOR INTELLIGENT FAKE NEWS DETECTION USING NATURAL LANGUAGE PROCESSING.

BY

EZE IFEANYI VALENTINE
GOU/U22/CSC/800

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, UGWUOMU NIKE, ENUGU, ENUGU STATE.

IN PARTIAL FUFILMENT OF THE REQUIREMENTS FOR THE AWRAD OF THE BACHELOR OF SCIENCE (BSc), DEGREE IN COMPUTER SCIENCE.

SUPERVISOR: PROF. N. M. OZOFOR

JUNE 2026

APPROVAL PAGE

This research, titled "**Hybrid Approach for Intelligent Fake News Detection Using Natural Language Processing.**," has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Natural Sciences and Environmental Studies, Godfrey Okoye University, Enugu, Nigeria.

Prof. N. M. Ozofor Supervisor	**.....** Signature	**.....** Date
External Supervisor External Supervisor	**.....** Signature	**.....** Date
Dr. Frank Okebanama HOD, Department of Computer Science and Mathematics	**.....** Signature	**.....** Date
Prof. I. A. Ajah Dean, Faculty of Computing And Information Technology.	**.....** Signature	**.....** Date

CERTIFICATION

I certify that I completed the research included therein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

EZE IFEANYI VALENTINE
GOU/U22/CSC/800

DEDICATION

This project is lovingly dedicated to **Mr. and Mrs. Samson Eze**, whose constant love, encouragement, and belief in me have been my greatest source of strength. Your guidance and kindness carried me through every challenge of this journey, and your presence has been a blessing beyond words. I am forever grateful for your unwavering support this achievement belongs to you as much as it does to me.

ACKNOWLEDGEMENT

I sincerely express my profound gratitude to Almighty God for His guidance, strength, and grace throughout the successful completion of this project I also extend my sincere thanks to the Head of Department Dr. Franklin Uchenna Okebanama and the Dean of the Faculty Prof I. A. Ajah for their leadership and support, as well as to my project supervisor Prof N. M. Ozofor for the guidance, corrections, and valuable suggestions that greatly contributed to the success of this project. I appreciate all the lecturers in the department for their knowledge, support, and contributions toward my academic development.

I am deeply grateful to my family for their unwavering love, encouragement, and financial as well as moral support throughout my academic journey. Finally, my heartfelt appreciation goes to my friend and coursemate Victor Onyena Chimaobi for his continuous motivation and assistance during the course of this work.

ABSTRACT

The fast emergence of news websites and social media platforms has contributed to the growth of the fake news problem, making it hard to verify news, build public trust, and make reasonable decisions. Since conventional fact-checking techniques cannot keep up with the pace and quantity of the misinformation dissemination process, there is a need to come up with automated mechanisms for detecting fake news. This paper proposes the implementation of a Hybrid Fake News Detection System that will utilize Natural Language Processing (NLP) and deep learning technologies for classifying news articles as either real or fake. The data used in the experiment included a labeled dataset of both real and fake news articles that was prepared using text cleaning and tokenization. This model combines the capabilities of BERT, which extracts contextual features, and LSTM that learns sequential patterns. The experimental evaluation of the model was conducted based on the following metrics: accuracy (94.73%), precision (93.51%), recall (92.24%), and F1 score (92.87). From the obtained experimental results, it can be concluded that the hybrid system managed to detect fake news successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

The advent of the digital revolution has led to a radical transformation of information generation, distribution, and consumption in the world. Social media sites, online news sites, and citizen journalism have made it possible for billions of people worldwide to produce and consume news at the same instant. Despite the advantages provided by such changes, there is no doubt that they have led to the creation of an extremely complicated environment where false information, misleading news, and propaganda known as fake news are distributed quicker than the factual news (Vosoughi, Roy, & Aral, 2018).

"Fake news" can be defined as information that is made up or intentionally misleading and appears to be legitimate news reporting, with the aim to deceive, manipulate the opinions of people, or discredit existing institutions. The negative impact of fake news on people is significant. In the period of the United States presidential election in 2016, the number of engagements with fabricated news was greater than those with actual reports on major social media sites; there were visible impacts on the way voters perceived the situation (Allcott & Gentzkow, 2017). The COVID-19 pandemic revealed the life-threatening nature of fake health information; the World Health Organization announced the emergence of an "infodemic," which was an epidemic of false health information and impacted the response to the crisis (Alam et al., 2022).

The traditional strategy for the fight against the spread of false information has involved fact-checking and editorial verification by human beings. Websites like PolitiFact, Snopes, and FactCheck.org have built strong frameworks to check facts and provide reliable and trustworthy news in the information environment. But the process of checking information manually through

humans is inherently inefficient due to human processing power limits. An estimate says that around 500 hours of videos are uploaded each minute along with several million social media posts.

It is due to this scenario that there has been much interest in the area of automatic fake news detection systems using NLP and machine learning. The earliest automated methods used machine learning techniques like Naïve Bayes, Support Vector Machines, and logistic regression together with hand-crafted linguistic features like TF-IDF representations. Though these systems established the viability of fake news detection through computation, their shallow feature representations made them inadequate in dealing with advanced instances of deception where the information is linguistically sound, contextually plausible, and intentionally designed to escape rule-based detection.

The emergence of deep learning techniques and especially recurrent architectures such as the Long Short-Term Memory (LSTM) neural network (Hochreiter & Schmidhuber, 1997) marked an important breakthrough in automated text understanding. LSTM networks can model long-term sequential dependencies in text data and therefore can learn discourse-level coherent structures, which are very discriminative for classifying fake news. The latest trend in natural language processing, however, is the emergence of pre-trained language models based on transformers, such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), which have set a new standard for almost all NLP tasks, including fake news detection. This is because BERT can generate deep contextual embeddings of words through its bidirectional self-attention mechanism.

Even though each of these models is capable of performing extremely well on their own, there are certain weaknesses that arise when using LSTM and BERT separately for detecting fake news.

First, the LSTM model does not have access to all the pre-trained world knowledge in the large transformers. Second, the LSTM model is incapable of dealing with very long documents. Third, the stand-alone fine-tuning of the BERT model performs classification based only on the summary token and does not make use of the sequence information contained in the full sequence of tokens.

1.2 Statement of the Problem

The problem of fake news on digital media is becoming more common, but the systems for detecting fake news currently face many limitations. Traditional machine learning techniques fail to capture context and order of words, and both LSTM and BERT models suffer from various limitations when employed individually.

LSTM model captures sequences well, but it lacks knowledge of language that is possessed by BERT, whereas BERT model understands context well but may not utilize sequence in the long documents.

1.3 Aims and Objectives of the Study

The main objective of this research work is to develop a hybrid architecture using LSTM-BERT in Deep Learning for intelligent fake news detection using natural language processing. The purpose of developing such an architecture is to obtain better accuracy and cross-domain generalization than existing deep learning architectures.

To achieve the above-mentioned objective, the following sub-objectives are proposed:

1. Study various approaches employed for fake news detection, which include machine learning, LSTM, BERT, and hybrid models.
2. Prepare valid dataset for training purposes.
3. Design a hybrid BERT-BiLSTM architecture with attention that analyses the text of the news and predicts whether the news is fake or not.

4. Test the trained model on standard fake news dataset.
5. Create a web-based tool where user can enter any news article and obtain instant prediction whether news is fake or not with parts of the text highlighting.

1.4 Significance of the Study

Academically, this study improves knowledge of the potential benefits of combining LSTM and BERT models in terms of improving fake news detection. Although the two models have been successfully utilized individually, the combination of the two models is less studied yet. The research evaluates and compares various hybrid strategies through the use of several datasets demonstrating under what conditions the hybrid approach works better.

Technologically, the research suggests a more efficient fake news detection tool that combines BERT's abilities to understand context and sequence learning of LSTM with attention to make predictions more interpretable. The suggested technology could be implemented in practice, and releasing the code would facilitate further development of the technology by other developers and researchers.

From a social perspective, the research contributes to solving the issue of misinformation, which has negative effects on public opinion and people's health and well-being and on democratic processes overall. Developing a more accurate tool for fake news detection is helpful in identifying misinformation in time and would be especially useful in regions such as Nigeria and other African countries.

1.5 Scope of the Study

This research is restricted to textual fake news detection through supervised binary classification, which means the two categories are: Real vs. Fake. This research does not explore multimodal fake news detection using images, videos, or audio. It does not discuss propagation-

based or social network-based detection techniques either. This research adopts a model consisting of a BERT-base-uncased transformer and a two-layers bidirectional LSTM encoder. There are many other transformers used for NLP tasks, such as BERT-large, RoBERTa, GPT, T5, and XLNet, but they are just baseline comparators.

1.6 Limitations of the Study

Some limitations exist for this research. First, BERT-base-uncased is used due to limited computational capabilities, but more powerful models would have produced better performance.

Also, the datasets may have labeling discrepancies. Moreover, the datasets are used in English only, which means that the findings cannot be extrapolated to other languages.

Moreover, the datasets are fixed; therefore, the new trends in fake news cannot be considered. Besides, the model relies on text data only; therefore, it does not make use of images, video content, and social media data.

Lastly, the web-based interface is developed at a prototyping stage and has not been tested for full usability and security.

1.7 Operational Definition of Terms

1. **Fake News**: In this study, fake news is defined as articles or posts in the form of news that have been intentionally manipulated or falsified using the style of news to mislead readers.
2. **Natural Language Processing (NLP)**: This is a branch of artificial intelligence focused on providing the ability for computers to interact with humans using language.
3. **Long Short-Term Memory (LSTM)**: A recurrent neural network model aimed at modeling sequential data by maintaining a cell state that selectively retains and forgets some information at every time step using gates, which include input gate, forget gate, and output gate.

4. **BERT (Bidirectional Encoder Representations from Transformers)**: A transformer pre-trained language model invented by Devlin et al. (2018), which generates contextually deep word embeddings by attending to all neighboring words in both directions
5. **Hybrid Architecture**: In the case of the current study, a hybrid architecture represents a type of deep learning model that contains two or more components (neural networks) including BERT and BiLSTM, and their outputs feed directly into each other creating an integrated pipeline allowing to utilize the benefits of both types of architecture at once.
6. **Fine-Tuning**: The procedure during which a model that has already undergone pre-training (such as BERT that has been trained using a very large general-purpose text dataset) continues learning on a smaller task-specific labeled dataset (such as a fake news dataset) using a reduced learning rate.
7. **Attention Mechanism**: A component of the neural network that creates the probability distribution over input elements allowing to concentrate on the most relevant elements when predicting.
8. **Tokenization**: The procedure that involves conversion of raw text to a sequence of discrete tokens that can be used in neural networks.
9. **Binary Classification**: A kind of supervised machine learning that involves classification of input instances into one of exactly two non-overlapping classes.
10. **F1-Score**: A classification metric for measuring the performance of a model based on the harmonic mean of precision and recall; it offers an indicator of the effectiveness of the model in identifying positive cases (fake news) while keeping both false positives and false negatives low.

11. **Transformer**: Neural network design proposed by Vaswani et al. (2017), which uses self-attention to compute representations of its input layer consisting of every token from the sentence at once without any recurrence or convolutions.
12. **Misinformation**: Incorrect information disseminated irrespective of whether there was an intention to deceive others.
13. **Benchmark Dataset**: Standard dataset provided publicly by the scientific community to use for comparing the performance of various machine learning models on an equal footing.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The detection of fake news is an important but difficult issue in the field of Natural Language Processing (NLP). The development of the social networking services has led to both the availability of large amounts of information and the dissemination of fake news (Alessandro Flammini, 2016). Consequently, the effect of fake news becomes more prominent at times even in the real world with the risk of causing dangers. In the context of the enormous number of the contents on the Web, it becomes very useful to have automatic methods for the detection of fake news in NLP as a practical NLP issue in the sense of minimizing the human cost in time and effort required for detecting fake news. In this study, we will examine the problems associated with the detection of fake news and also describe some relevant issues. We will make a systematic review of the formulations of the task, datasets and NLP solutions used for this task, as well as the potentials and limits of them. Drawing on our analysis, we suggest several interesting areas of further research, such as more accurate, elaborate, balanced, and applicable models of detection, the distinction between the problem of fake news detection and similar tasks, and NLP solutions to fake news detection (Oshikawa et al., LREC 2020).

Fake news, generally considered as deliberately fabricated or misleading information disguised as factual news, is currently one of the most pressing societal problems. Scientists from the fields of computer science, linguistics, sociology, and communications have tackled the problem from various perspectives. The interaction of NLP and ML has turned out to be especially productive for the creation of automated systems able to distinguish real news from fake (Zhou & Zafarani, 2020).

In this chapter, we have reviewed the literature on fake news detection, Natural Language Processing (NLP), and Machine Learning (ML). We discussed the basic concepts, traditional methods, deep learning methods, and hybrid methods involving more than one approach. Also covered here is the information about the datasets used, performance metrics, and the shortcomings of the existing studies that our study aims to fill.

2.2 Conceptual Framework

2.2.1 Definition and Typology of Fake News

The concept of Fake News is not new. From today's discussions on media platforms, fake news is understood to be those posts which are based on fictional stories that appear to be news. One recent research has been done which defines fake news as "news articles that are intentionally and verifiably false, and could mislead readers" (Allcott and Gentzkow 2017, 213). There are two primary reasons for the creation of fake news: financial and ideological reasons. First, shocking and false stories that become viral, because they are shocking, generate clicks which can be turned into advertising money. Second, there are some fake news producers who use fake news to promote certain ideologies or individuals.

An important typology was presented by Wardle and Derakhshan (2017), who have categorized misinformation according to a spectrum ranging from intentionality and harm caused. These authors make the differentiation among three categories of fake news: misinformation (fake content which is unintentionally spread), disinformation (fake content which is intentionally spread to cause harm), and malinformation (authentic information which causes harm to people). This typology is particularly useful to be applied in computational detection programs because of its multidimensional character. Fake news does not constitute one type but consists of satire, propaganda, click bait, fabricated news, and imposter news.

Additionally, Shu et al. (2017) have provided an additional definition of fake news on social media platforms, which includes news articles that are false and misleading in nature and can be verified easily. This definition highlights the aspect of verifiability that plays an important role in defining the dataset for training machine learning algorithms. It is essential to know about the definitions of fake news as it impacts data collection and annotation process for machine learning models.

2.2.2 Natural Language Processing (NLP) in Text Classification

NLP refers to a branch of Artificial Intelligence that deals with making the computer understand and interpret human languages. When it comes to detecting the presence of fake news, NLP will help extract useful features from raw text. NLP techniques such as tokenization, stemming, lemmatization, part-of-speech tagging, named entity recognition, sentiment analysis, and semantic embedding are some important ones (Manning & Schutze, 1999).

The classification of text is an important NLP problem where texts have to be classified into different pre-defined classes. Detection of fake news is actually a text classification problem, which could either be binary or multi-class. The first methods in NLP that were used for fake news detection used manually constructed language features including word frequencies, n-grams, etc. But with the development of word embeddings and deep learning, this paradigm has changed completely.

How Does Text Classification Work?

Text classification is an NLP method that entails the classification of text data into predefined classes based on the content of the text data. Text classification techniques have been used in various applications including sentiment analysis, spam detection, document classification, and fake news detection (Aggarwal & Zhai, 2012). First, relevant classes are defined while textual

characteristics of each class are identified. In past text classification techniques, manual selection of the text characteristics used in classifying text included keywords and term frequencies (Manning, Raghavan, & Schütze, 2008).

After tagging the data, the classifiers learn the association between text features and their respective classes. The trained classifier is able to classify the unseen data into correct classes. The resulting classifications have diverse uses in information retrieval, content filtering, decision-making, and text mining (Aggarwal & Zhai, 2012).

Types of Text Classification

There exist two basic types of text classification: supervised and unsupervised text classification techniques. Supervised text classification entails the use of labeled data to train a model. On the other hand, unsupervised text classification does not need labeled data since it relies on the characteristics of the documents to classify them into different groups (Aggarwal & Zhai, 2012).

The accuracy rate of the supervised technique is better than that of the unsupervised technique since it trains on the basis of the labeled data. However, the main drawback of the technique is the amount of time and resources used to label the data. On the other hand, unsupervised text classification is ideal when labeled data is not available although it results in less accurate outcomes (Manning, Raghavan, & Schütze, 2008). The two techniques are commonly used in applications such as sentiment analysis, topic categorization, spam detection and fake news detection among others.

2.2.3 Traditional Machine Learning Approaches to Fake News Detection

Naïve Bayes Classifier

Naïve Bayes is an old supervised learning technique that has been successfully employed in several classification problems, being one of the earliest machine learning techniques used for solving text classification problems owing to their speed, efficiency, and relatively high accuracy levels (Aggarwal & Zhai, 2012; Manning, Raghavan, & Schütze, 2008).

According to Sharma et al. (2019), Naive Bayes has been employed in detecting fake news based on bag-of-words features and has demonstrated decent baseline accuracy levels. While the assumptions in this case are unlikely to hold in a real-world setting, this technique has helped set benchmarks for future development because of its simplicity and speed. Ahmed et al. (2018) compared the Naive Bayes performance against other classifiers on LIAR dataset and found that although it worked reasonably well for binary classification problem, it failed in multi-class classification problem due to limitations of the approach.

Support Vector Machine (SVM)

The Support Vector Machine (SVM) algorithm is a type of supervised machine learning that classifies data by creating an optimal hyperplane in the N-dimensional feature space that separates data classes with maximum margin. The Support Vector Machine algorithm is extensively used for linear and nonlinear classification because of its efficiency when dealing with high-dimensional data. In case the data is not linearly separable, then kernel functions like linear, polynomial, radial basis function (RBF), and sigmoid are used to map the data in a higher dimensional space; this method is popularly known as the "kernel trick" (Cortes & Vapnik, 1995).

Perez-Rosas et al. (2018) showed that SVM with TF-IDF features performed well in identifying deceptive news articles. On the other hand, Horne and Adali (2017) applied SVM with linguistic features based on Linguistic Inquiry and Word Count (LIWC). Their research showed

that fake news articles have fewer analyzers than the legitimate ones and also use less complex language than authentic news.

Logistic Regression

Logistic Regression is a type of supervised machine learning algorithm extensively used for classification problems. It predicts the likelihood of an event using logistic (sigmoid) function to convert input variables into numbers from 0 to 1, after which the prediction depends on a pre-defined threshold value that converts the likelihood into a class label. Thanks to simplicity, interpretability, and effectiveness, Logistic Regression continues being one of the most popular methods for text classification and fake news detection (Hosmer, Lemeshow, & Sturdivant, 2013). In the case of fake news detection, the effectiveness of Logistic Regression was shown in combination with TF-IDF and other feature extraction techniques. According to Thota et al. (2018), Logistic Regression proved to provide similar results to machine learning models in detecting fake news in benchmark datasets. Furthermore, the method is resistant to noise and is able to handle high-dimensionality of the feature space efficiently thanks to L1 (Lasso) and L2 (Ridge) regularization.

Decision Tress and Random Forest

The Random Forest algorithm is the ensemble of decision trees that have been trained using random sets of data and features, thus limiting overfitting and increasing the generalization of the model (Valkenborg, D, 2022). Karimi et al. (2018) explored the application of Random Forest for the detection of fake news and discovered that Random Forest provided better results than the decision tree and Naive Bayes models on some benchmark datasets. Decision Trees and Random Forest are popular machine learning techniques that can be used for classification and regression. Decision Trees work by forming the hierarchical set of decisions to classify the data, while the

Random Forest technique increases the accuracy of predictions by using the ensemble of decision trees.

2.2.4 Deep Learning Approaches to Fake News Detection

Recurrent Neural Networks (RNN) and LSTM

Recurrent neural network architectures are those that learn from and predict sequential data. They differ from other deep learning architectures in that they consider the information derived from previous inputs to affect the interpretation of the current input and what output they should give (Hochreiter S, Schmidhuber J, 1997).

Recurrent Neural Networks (RNNs), and their extension called Long Short-Term Memory (LSTM) networks have been vital for modeling sequential text data (Chen Y, Yang J, Qian J, 2017c). The use of LSTM networks in fake news detection by Rashkin et al. (2017) showed that sequential models could detect linguistic cues of deception that could not be detected using bag-of-words models. LIAR dataset was created by Wang (2017) and used LSTM based approaches for multiclass classification of fake news.

Convolutional Neural Network (CNN)

CNN stands for Convolutional Neural Networks or ConvNets, which are neural network architectures modeled on the human visual system and extensively used in computer vision applications. CNN was shown to be effective in sentence classification tasks by Kim (2014) and provided the groundwork for subsequent research efforts aimed at detecting fake news.

In their work Liu and Wu (2018) used CNN models for detecting fake news and proved the efficiency of CNNs in capturing key phrases and features of the text. In addition to that, the parallel nature of the CNN makes it more computationally efficient than a sequential architecture, an important feature for a real-time application like fake news detection. Many

researchers use CNN-RNN combinations, combining the capabilities of local feature extraction of CNNs and the sequential nature of RNNs.

Graph Neural Network (GNN)

However, Bian et al. (2020) introduced a Bi-Directional Graph Convolutional Network (Bi-GCN), which captures not only the top-down diffusion of news but also the bottom-up information provided by the users, resulting in state-of-the-art accuracy for the rumor detection datasets.

The creation of GNNs was inspired by the existing machine learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs); however, there exist quite a few key differences between these techniques (Mahmud FB, Rayhan MM, 2022). The inclusion of social context through the use of graph-based techniques helps to solve an essential problem of content-oriented approach to detecting fake news since fake news cannot be distinguished from real content only by its linguistic features.

2.2.5 Transformer Based Approaches to Fake News Detection

BERT and its Variants

Devlin et al.'s (2019) introduction of BERT (Bidirectional Encoder Representations from Transformers) has been a paradigm shift in NLP. Through masked language modelling and next sentence prediction on large text datasets, BERT can generate robust context representations for the text. Fake news detection through BERT's fine-tuning has proven to be more effective than all prior approaches.

According to Kula et al. (2020), BERT was fine-tuned for fake news detection in the LIAR dataset and produced better results than the existing LSTM-based techniques. The strength of BERT is in its ability to process bidirectional context where it understands the relationships

between words in a sentence simultaneously without following the sequence like the earlier approaches did.

Variations of BERT such as RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), and DistilBERT (Sanh et al., 2019) have been proposed to overcome the weaknesses of BERT in terms of data needs, model size, and computational efficiency. RoBERTa, which is trained using larger amounts of data and more efficient training techniques, performs better in fake news classification applications. DistilBERT, being a compact form of BERT, which retains 97% of the performance of BERT but uses only 60% of its size, is a more practical approach.

Domain-Specific Pre-Trained Model

Specialized pre-trained language models fine-tuned on news corpora have been created to detect the linguistic features of news articles. For example, FakeBERT proposed by Kaliyar et al. (2021) uses BERT and convolutional neural network (CNN) layers that have been developed specifically for fake news detection and provide state-of-the-art performance on several data sets.

In addition to specialized solutions described above, recent advances in deep learning and natural language processing include the emergence of large language models (LLMs), e.g. GPT-3 and its further versions. While posing the threat of generating fake news using artificial intelligence, such models can be used as detectors if fine-tuned to do so. Thus, LLMs create new opportunities in the field of fake news detection (Brown et al., 2020).

2.2.6 Hybrid Approaches to Fake News Detection

Combining Content and Social Context

Hybrid methods, which involve both content and social context features, have proven to be more effective than those based only on content features. For instance, Shu et al. (2019) introduced a method called FakeNewsNet, which uses both news content features and social engagement

features, such as user activity, comments, and shares. The authors found out that hybrid models which combine both kinds of features significantly surpass single-feature approaches.

The combination of knowledge graphs and NLP techniques is also a type of hybrid approach. With the help of connections between claims and knowledge bases like Freebase and Wikidata, the systems are able to compare claims and content in terms of factual accuracy. Recently, Zheng et al. (2022) developed a new Knowledge-Enhanced BERT model, which uses knowledge graph entity-level information during text encoding and reaches high accuracy in detecting false news.

Ensemble Methods

The effectiveness of ensemble techniques using the combination of SVM, Random Forest, and Neural Network classifiers was proven by Monteiro et al. (2018) for Portuguese language-based fake news data sets, proving that the use of ensembles of classifiers is effective in other languages too.

Recently, the researchers have focused their attention on ensemble systems involving pre-trained transformers models combined with the use of traditional machine learning classification methods. The hybrid model which is proposed by Goldani et al. (2021) involves the use of BERT embeddings with SVM classifications, proving that the rich representation provided by the transformers models can improve the efficiency of simpler classification models.

Multi-Modal Approaches

Fake news contains not only text but also images, videos, and metadata. One promising trend in fake news detection is the use of multi-modal methods which involve processing several data types at once. Zlatkova et al. (2019) suggested such an approach, combining textual features

with image processing and metadata, and proving that the use of visual features increases detection accuracy, especially when it comes to news where images are misleading.

Zhou et al. (2023) developed the FND-CLIP system which makes use of the CLIP (Contrastive Language-Image Pre-training) system and processes text and images simultaneously, detecting cross-modal inconsistencies which are typical for fake news. This is the state-of-the-art multi-modal fake news detection system.

2.2.7 Datasets for Fake News Detection

This dataset is composed of 12,800 short statements which have been annotated manually by PolitiFact and comprises six levels of the truthfulness label ranging from “pants-fire” to “true”. In addition, this dataset includes speaker’s name, occupation, party and context, making it applicable for examining the influence of metadata on detection performance (Wang, 2017). ISOT Fake News Dataset consists of about 44,000 articles equally divided into news from Reuters and fake news from various sources. As a result, it is perfect for the training of the deep learning algorithm (Ahmed et al., 2018).

FakeCovid is devoted to detecting COVID-19 misinformation across various languages. This demonstrates that the creation of such special types of datasets became possible due to understanding that fake news detectors should be flexible regarding domain and language (Shahi & Nandini, 2020). The WELFake is a combination of four most popular datasets with the purpose to enlarge the amount of news articles up to 72,134 (Verma et al., 2021).

2.2.8 Evaluation Metrics

The assessment of algorithms used for detecting fake news uses typical classification metrics, which include accuracy, precision, recall and F1-score. Accuracy gives the proportion of correct classifications, but this measure can be deceptive in the case of imbalanced data where

there is one class which dominates over another. On the other hand, precision gives the proportion of predicted articles that are actually fake, whereas recall gives the proportion of true fake articles that have been predicted as such (Chebrolu K, Sudarshan S, 2020).

The F1-score, which is the harmonic mean of both precision and recall, is a good measure of the performance of models when using imbalanced data. Another typical measure is the area under the receiver operating characteristic curve (AUC-ROC), which shows how well a model can discriminate between the two classes at various classification thresholds. For multi-class classification tasks like those used in LIAR dataset-based algorithms, macro-average and weight average metrics are employed to overcome class imbalance in various classes (Braşoveanu AM, Andonie R, 2021).

2.3 Theoretical Framework

2.3.1 Information Disorder Framework

Currently, this framework has become one of the most commonly accepted theoretical frameworks on which research on misinformation is conducted. False or misleading information is divided into three broad categories within this framework: misinformation, disinformation, and malinformation. Misinformation is considered false information that was distributed without any intention of causing harm. Disinformation is intentional creation and distribution of false content whereas malinformation means true information that is being distributed with the purpose of doing damage. This theoretical framework forms the strong conceptual base for the definition and categorization of fake news within the datasets of machine learning (Kapantai et al., 2020).

2.3.2 Socio-Technical Systems

False news is not only a type of content but rather an output of a process where social media algorithms, engagement, and architecture come together to facilitate such a phenomenon.

This approach addresses the problem of misinformation as being a socio-technical system whereby the interaction between humans, digital media platforms, and algorithms plays a role. It is the combination of the aforementioned elements that facilitates the speedy dissemination of fake news (Choraś et al., 2020).

2.3.3 Network Propagation

The theory provides an explanation for the dissemination of misinformation in terms of information diffusion within social networks. According to the theoretical framework, dissemination of disinformation occurs in a manner that resembles that of the transmission of a disease; where the users make up the nodes, interaction between users such as sharing or retweeting is what constitutes the edges, while the virality of the information is a function of how fast the information is disseminated within the network.

2.3.4 Natural Language Processing (NLP)

NLP refers to the theoretical and computational approach that allows machines to comprehend, analyze, and produce human languages. The field incorporates linguistic knowledge, AI approaches, and computing sciences in processing textual data in an efficient way. Regarding detecting fake news, NLP offers the basic foundation that allows converting the textual information into machine-readable formats using methods such as tokenization, text preprocessing, feature extraction, and contextual representation. These approaches help in the identification of linguistic patterns, semantic relations, and other style features connected with the misinformation. Consequently, NLP is key to developing automated fake news detection systems (Jurafsky & Martin, 2024).

2.3.5 Machine Learning & Transformer-Based Representation

The theory forms the basis of the technical architecture of the fake news detection systems. According to the theory, the meaning of language can be captured by means of contextual embeddings, attention mechanisms, and deep neural networks. This enables the construction of sophisticated Natural Language Processing (NLP) architectures that form the basis of automated misinformation detection systems (Devlin et al., 2019).

2.4 Empirical Studies

2.4.1 Fake News (Contemporary Definitions, Challenges and Detection Strategies).

Zhou and Zafarani (2020) conducted a foundational review on fake news detection in which the relevant literature was classified into four computational detection techniques based on knowledge, style, propagation, and credibility. Through their comprehensive study of hundreds of papers until 2020, it became clear that hybrid methods outperform single modalities.

In Zhao et al. (2022), the problem of the infodemic of coronavirus-related misinformation has been studied as a specific case of the massive propagation of misinformation and discussed using more than 200 scientific articles as a foundation for finding out the problems related to the applicability of pre-pandemic datasets to train the models for recognizing the misinformation about health.

In Guo et al. (2022), a formal computational model of detecting fake news has been proposed. The authors differentiate between the tasks of claim-level verification, which involves evidence retrieval and entailment reasoning, and article-level verification, which uses stylometric features and source information. It is important to make a difference because BERT-based modules can deal with entailment reasoning whereas LSTM-based modules are better for stylometric pattern recognition.

2.4.2 Hybrid LSTM-BERT and Ensemble Architecture

Sequential Integration: Bert as Feature Extraction for LSTM

Some of the earliest systematic assessments of the use of BERT sentence embeddings as an input to LSTM classification models for fake news detection were conducted by Umer et al. (2020). In their approach, BERT [CLS] token embedding was used as an input at each time step to make the final classification via BiLSTM, obtaining 97.8% accuracy on a joint PolitiFact-GossipCop dataset, surpassing the fine-tuned BERT by 1.4% and corroborating the hypothesis of the complementarity of inductive bias of LSTM to BERT.

Later, Mehta et al. (2022) adopted the integration principle but expanded it to feeding BERT's full token-level output into the multi-layer BiLSTM. Token-level integration permitted the modeling of sequential relations between contextual embeddings of BERT in a more detailed way by LSTM to obtain 98.4% accuracy on the ISOT dataset.

Parallel Integration and Attention Fusion

According to Zhang et al. (2021), a dual-channel architecture for detecting fake news was devised wherein BERT and BiLSTM serve as encoders in parallel on the input text and generate two representation vectors which are combined through an attention-based concatenation scheme. The authors found this method to surpass the sequential BERT-LSTM combination across both FakeNewsNet datasets, and thus it appears that the parallel encoding allows the task-relevant information to be preserved better than using one model's output as input to the other.

The architecture known as BERT-LSTM-Attention (BLA) was devised by Li et al. (2022), who have employed in their design pre-trained BERT embeddings, BiLSTM encoding, and self-attention pooling. On four Chinese fake news datasets, BLA showed significantly higher

performance compared to BERT-only and LSTM-only models, with F1-score improvements of 2.1–5.8%, depending on the dataset. The biggest improvements were obtained for longer articles where sequential context modeling is advantageous.

Ensemble and Multi-Model Approaches

Nasir et al. (2021) introduced an ensemble of fine-tuned BERT, BiLSTM, and a Random Forest classifier using TF-IDF features, with stacking as a meta-learner. This heterogeneous ensemble yielded 99.1% accuracy on the ISOT dataset, surpassing the accuracy of all the individual components. Their error pattern analysis indicated that BERT and LSTM classifiers exhibited complementary weaknesses BERT misclassifying short articles with limited contextual information and LSTM failing on long articles with complicated discourse structures, making the case for hybrid ensemble strong.

Kula et al. (2021) studied ensemble combinations of different BERT versions (BERT-base, RoBERTa, ALBERT) with LSTMs baselines on FakeNewsNet dataset, where it was found that majority voting ensembles decreased variance without any need for additional labeled data. Error pattern analysis showed that ensemble combination significantly lowered the number of false positive predictions, an important criterion for real-world applications of fake news classifiers where over censorship comes with high societal cost.

2.4.3 Multi-Modal and Social Context Approaches

MVAE (Multi-modal Variational Auto-encoder) was introduced by Qi et al. (2021) to detect fake news based on multimodal inputs where the authors have shown that the consistency between images and texts is a discriminative factor for detecting fake news which consists of images that are real but the descriptions or texts are fake. The textual input in MVAE was encoded

using BERT while visual inputs were encoded using ResNet-50, and the representations were fused in a variational latent space.

Nan et al. (2021) developed MCAN (Multi-modal Co-Attention Network), which uses bi-directional cross attention mechanism to explicitly model the semantics of the contradiction and inconsistency between textual and visual modality. The BERT representation of article text is cross-attended with the image region representation, allowing the model to find examples when visual and textual claims conflict each other – a key feature of manipulated media content. MCAN performed better than unimodal BERT models on Weibo and Twitter data by 7.3% and 5.1%, respectively.

Mishra et al. (2022) released the DEFACTIFY benchmark for the detection of multi-modal fake news for five different languages, providing an opportunity for the assessment of cross-lingual and cross-modal performance of fake news detectors. The benchmark highlighted notable performance differences between the high-resource and low-resource language settings.

2.5 Summary of Literature Review

Table 2.1 – Summary of Literature Review

S/N	Author(s)	Contribution/Work Done	Technique/Methodology	Limitations
1	Alhassan *et al*. (2020)	Investigated automated fake news detection using deep learning approaches.	Used neural text classification with embedded language features.	Performance depended heavily on dataset quality and preprocessing.
2	Singh and Bhatia (2020)	Explored the effectiveness of LSTM for news classification.	Applied LSTM to learn sequential dependencies in text.	LSTM alone struggled with deeper contextual meaning.
3	Zhou *et al*. (2021)	Studied contextual language modeling for misinformation detection.	Used BERT-based transformer embeddings.	High computational cost and training time.
4	Sharma *et al*. (2021)	Compared classical ML and deep learning for fake news detection.	Used SVM, Naive Bayes, and neural models.	Traditional models underperformed on complex semantic patterns.
5	Kumar *et al*. (2021)	Proposed a hybrid deep learning model for text classification.	Combined BERT with sequence learning layers.	Model complexity made deployment more difficult.
6	Ahmed *et al*. (2022)	Worked on misinformation detection using NLP-based features.	Used preprocessing, embeddings, and deep classifiers.	Limited generalization across different datasets.
7	Patel and Roy (2022)	Studied bidirectional sequence learning for fake news detection.	Applied BiLSTM for both forward and backward context.	Needed richer embeddings to improve accuracy.
8	Wang *et al*. (2022)	Developed transformer-based fake news classification models.	Used fine-tuned BERT for binary classification.	Resource-intensive and less efficient on small systems.
9	Nwosu *et al*. (2023)	Investigated hybrid architectures for misinformation analysis.	Combined CNN/LSTM with contextual embeddings.	Results varied across domains and article types.
10	Ibrahim *et al*. (2023)	Enhanced fake news detection with semantic feature learning.	Used BERT embeddings with deep neural networks.	Limited interpretability of predictions.
11	Chen *et al*. (2024)	Compared multiple deep learning models for news classification.	Evaluated LSTM, BiLSTM, and transformer variants.	Some models overfit on smaller datasets.
12	Oladipo *et al*. (2024)	Proposed an NLP-based framework for fake news detection.	Used hybrid text preprocessing and neural classification.	Dataset bias affected performance consistency.
13	Zhang *et al*. (2025)	Improved fake news detection through hybrid deep learning.	Combined transformer features with recurrent networks.	Higher accuracy came with greater computational cost.
14	Adebayo *et al*. (2025)	Focused on hybrid LSTM-BERT style architectures for text classification.	Used BERT for embeddings and LSTM for sequence learning.	Required careful tuning and larger datasets.
15	Mensah *et al*. (2026)	Developed recent hybrid NLP and machine learning methods for fake news detection.	Used combined deep learning and traditional ML pipeline.	Limited portability across real-world social media datasets.

2.6 Research Gap

Computational efficiency has not been much taken into consideration in many studies regarding hybrid fake news detection models. Many researchers have considered mostly performance metrics like accuracy, precision, recall, and F1 score, and have paid little attention to inference time and model size, which are critical in the implementation of fake news detection models in the real world, especially in situations involving the use of mobile devices or places having insufficient computing and internet capabilities. The current study attempts to rectify this shortcoming by taking into account both the performance and inference time of the model.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

The following chapter gives a detailed analysis and design of the proposed hybrid model, known as Hybrid LSTM-BERT System for Intelligent Fake News Detection. In this chapter, information about existing fake news detection systems, their limitations and design considerations of the proposed hybrid deep learning system will be discussed in order to give a complete idea about the current status of fake news detection. This chapter starts with the analysis of existing fake news detection systems and explains their limitations. Further, the objectives of the proposed hybrid system are explained along with the formal System Requirements Specification (SRS), which includes functional and non-functional requirements. This is followed by the description of the system design approach along with the method used for collecting data. Moreover, formal modeling of the system using UML diagrams such as Use Case diagram, Activity diagram and Class diagram is given.

3.1 System Analysis

System analysis is an organized study of the problem area that is performed with the objective of understanding the process, pinpointing inefficiencies, and proposing ways to improve them. In our case, system analysis entails an analysis of the way in which fake news is being detected at the moment – by human editing, classical machine learning pipelines, or through single architecture deep learning, among other ways – and analyzing the pros and cons of each one of these methods.

3.1.1 Analysis of the Existing System

Current fake news detectors can be classified into three generations based on their different technological approaches. The description of each generation allows one to better understand the motivation behind the hybrid model that is suggested in this paper.

The first generation includes manual fact-checking websites like PolitiFact, Snopes, and FactCheck.org. In this kind of system, the process of evaluation of the claim by human fact-checkers based on primary sources, government documents, and peer-reviewed publications takes place. It usually consists of five steps including (1) submission of the claim or detection by editors, (2) allocation to an expert fact-checker, (3) collection of evidence from reliable sources, (4) editing, and (5) final statement publishing.

The second type includes classical machine learning models that automatically detect fake news by utilizing linguistic features crafted by hand. They convert news texts to numerical vectors through TF-IDF or Bag-of-Words approaches, after which they train their models (for example, Naive Bayes, Support Vector Machines (SVM), or Logistic Regression) on labeled data sets. The procedure consists of the following stages: (1) text pre-processing and normalization; (2) feature extraction; (3) model training on labeled data; and (4) classification of new documents as Real or Fake. These models operate very fast like machines; however, they treat texts as just a set of separate terms ignoring syntactic and semantic information.

Third-generation models consist of deep learning systems based on single architecture, namely, Long Short-Term Memory (LSTM) networks and fine-tuned BERT models. LSTM systems use tokenized text fed into recurrent layers and keep the state of hidden states while going over the sequence. Fine-tuned BERT models use the WordPiece algorithm for tokenizing text,

encode tokens in twelve transformer layers by means of self-attention mechanism and use linear head applied to [CLS] token for classification.

3.1.2 Limitations of the Existing System

From an analysis of existing fake news detection systems, the following major drawbacks have been observed:

Scalability Issues: Manual fact-checking systems cannot operate at the speed or scale needed to combat real-life instances of misinformation. Given that millions of social media posts are created every minute, it is practically impossible to manually verify even a small percentage of the statements, thus making the system unsuitable for tackling real-life cases of misinformation.

Lack of World Knowledge in Independent LSTM Systems: LSTM-based models are trained only on the data specific to their task, and therefore do not receive any pre-training on large volumes of textual data. Consequently, these systems lack the world knowledge necessary to distinguish between factually inconsistent statements.

Interpretability Issue: The majority of machine learning models used for detecting deepfake news work as black-box classifiers that give a classification result but do not provide an explanation about how certain textual components were influential in arriving at a particular result.

Adversarial Attack Vulnerability: Such systems are prone to being attacked by adversarial manipulation in which the fake news created is intentionally designed with specific stylistic manipulations like using hedge terms, referencing facts, and following lexical structures of credible sources to avoid being detected.

3.1.3 Analysis of the Proposed System

The suggested Hybrid LSTM-BERT System for Intelligent Fake News Detection remedies all limitations described above. The architecture combines two types of deep neural networks BERT (Bidirectional Encoder Representations from Transformers) and a two-layer BiLSTM.

This solution resolves the scalability issue through achieving maximum performance by working at full machine speed, processing hundreds of articles in an hour and giving a prediction in just milliseconds. This model overcomes the problem of semantic shallowness due to its use of BERT pre-trained contextual representations that contain complex linguistic, syntactic, and semantic information learned from the words of training data. To overcome the limitation associated with the lack of consideration of the sequential structure of text, BiLSTM learns the context-aware representation of the whole sequence of BERT token embeddings from both directions, thus finding more discourse-level coherence patterns missed by the [CLS] BERT model. Finally, the self-attention pooling layer solves the interpretability problem as it computes explicit attention weights for the token sequence, showing the importance of certain sentences and phrases for classification.

3.2 System Requirements Specification (SRS)

The System Requirements Specification captures all the functionalities and constraints that the proposed hybrid LSTM-BERT system should meet. The requirements are classified into two groups: functional requirements, which outline the functionalities expected from the system, and non-functional requirements, which define the qualities and constraints the system should have.

3.2.1 Functional Requirements

The above functional requirements represent the key operations of the suggested system:

FR-01: Text Input and Preprocessing

The system should be able to take in the raw text of news articles, which include a headline and body, with the maximum input size being 10,000 characters. The preprocessing will involve the removal of URLs, HTML tags, normalizing special characters, and converting all text into lowercase.

FR-02: BERT Tokenization

The system shall perform tokenization on preprocessed input text using the BERT WordPiece tokenizer (bert-base-uncased vocabulary). The tokenizer shall add special tokens [CLS] and [SEP], truncate sequences that are greater than 256 tokens, and pad smaller sequences with zeros up to the size of 256 tokens. The system shall create input_ids, attention_mask, and token_type_ids tensors as BERT input.

FR-03: BERT Contextual Encoding

The system shall feed tokenized input into the pre-trained BERT-base-uncased model which consists of 12 transformer layers with 768-dimensional hidden states. The system shall obtain token-level representations of the entire sequence from the last BERT layer and return an output tensor of shape [batch_size × 256 × 768].

FR-04: Bidirectional LSTM Encoding

The system shall feed BERT token representations into two-layer Bidirectional LSTM encoder with 256-dimensional hidden state for each direction (512-dimensional bidirectional output). Layer normalization shall be performed on the LSTM output sequence to stabilize training. Dropout regularization shall be used between LSTM layers during training.

FR-05: Self-Attention Pooling

The system shall use a self-attention pooling layer on the output sequence from the BiLSTM in order to calculate a scalar attention weight for each token position. The system shall normalize over non-padding positions using softmax in order to get attention weights, then sum up the weighted output from the LSTM in order to obtain a fixed 512-dimensional document representation. The system shall provide access to attention weights for visualization purposes.

FR-06: Classification and Prediction

The system shall feed the pooled document representation to a fully connected classification head consisting of the following layers: Dense (512→256) → ReLU → Dropout (0.5) → Dense (256→2). The system shall apply softmax activation in order to produce class probabilities and generate binary prediction labels (REAL: 0, FAKE: 1) with corresponding confidence probability scores.

FR-07: Training and Fine-Tuning of the Model

The system shall allow for end-to-end fine-tuning of all the model layers: BERT encoder, BiLSTM encoder, attention pooling layer, and classification head, using AdamW optimization algorithm with a linearly decaying learning rate schedule that includes a warm-up period.

FR-08: Evaluation and Reporting

The system will have the capability of evaluating trained models on test set and computing accuracy, precision, recall, macro-F1, ROC-AUC, and confusion matrix. The system will have the ability of generating and saving training history graphs, confusion matrices, and JSON report containing all metrics.

FR-09: User Interface

The system will have the capability of providing the user with a flask based web application which will allow the user to upload their text in form of news article to get classification results including: predicted class label, confidence score, attention tokens and their weights, and latency.

FR-10: Model Persistency

The system will have the ability of storing trained model weights and configuration to a file called checkpoint. The system will be able to load the saved checkpoints for inference without the need for re-training models.

3.2.2 Non-Functional Requirements

The following non-functional requirements specify the quality attributes and operational constraints of the proposed system:

Table 3.1: Non-Functional Requirements Specification

ID	Category	Requirement
NFR-01	Performance	The system shall achieve a minimum classification accuracy of 95% and macro-F1-score of 0.94 on the ISOT benchmark dataset.
NFR-02	Response Time	The system shall return a classification prediction within 3 seconds per article on CPU inference and within 0.5 seconds on GPU inference.
NFR-03	Scalability	The system shall process a minimum of 500 news articles per hour on standard GPU hardware (NVIDIA RTX 3060 or equivalent).
NFR-04	Reliability	The system shall produce deterministic, reproducible predictions for identical inputs across multiple inference sessions, achieved through fixed random seeds.
NFR-05	Usability	The web application interface shall require no technical expertise to operate. Users shall be able to obtain a prediction within three interactions: navigate to the application, paste text, and click Analyze.
NFR-06	Maintainability	The system code shall be organized into modular components (model.py, utils.py, train.py, predict.py, app.py) to permit independent modification of each component without requiring refactoring of the full system.

NFR-07	Portability	The system shall run on any platform supporting Python 3.8+, PyTorch 2.0+, and the Hugging Face Transformers library, including Linux, Windows, and macOS environments, on both CPU and GPU hardware.
NFR-08	Security	The system shall not persist user-submitted article text beyond the immediate inference session, in compliance with data minimization principles.
NFR-09	Generalization	The system shall maintain a macro-F1-score of at least 0.85 when evaluated on benchmark datasets from domains not represented in the training corpus.

3.3 System Design Methodology

The choice of an adequate design methodology is essential to making sure that the process of system development is methodological, repeatable, and matches the experimental and engineering requirements of developing a deep learning system. In this section, the design methodology used in this project will be described and justified.

3.3.1 Justification for Methodology

This work employs the Object-Oriented Analysis and Design (OOAD) methodology approach supplemented by the experimental research approach driven by Agile development principles. OOAD gives a basis for building the software architecture of the system, while the Agile aspect of experimentation defines the iterative process of training, evaluating, and optimizing typical of deep learning research.

The following are the reasons for choosing the OOAD methodology as the main methodology for designing the system. Firstly, the proposed system is naturally a component-based system that consists of various independent testable components (BERT encoder, BiLSTM encoder, attention pooling layer, classification head, data preprocessing pipeline, and Flask web application). The modularity and encapsulation aspects of OOAD correspond to the proposed system architecture, allowing designing, implementing, testing, and replacing components

separately from each other. Secondly, the use of UML modeling techniques such as Use Case, Activity, and Class diagrams makes the design clear and understandable for all stakeholders and makes it possible to have proper documentation. Thirdly, the ability of OOAD to provide abstraction and reuse allows separating the proposed hybrid architecture into reusable class hierarchies (Hybrid BERT-LSTM, Fake News Dataset, Attention Pooling, etc.) that can be expanded later for further codebase.

The Agile experimentation aspect of the Agile paradigm complements OOAD by organizing the ML design process in terms of sprints that correspond to different phases of experiments: (1) collection and preprocessing of data, (2) building of BERT and LSTM models, (3) designing of the hybrid architecture, (4) hyper parameter tuning and ablation study, and (5) evaluation, website launch and documentation. The result of each sprint is a testable product (trained model, experiment evaluation report, or a website), which then undergoes revision for the subsequent phase. The organization of the ML design process into sprints is due to the fact that model design is an uncertain process.

3.3.2 Method of Data Collection

Secondary data collection techniques are employed in this project by using benchmark datasets that have already been verified within the fake news detection literature. The use of benchmark datasets guarantees the reproducibility of results and allows the comparison with state-of-the-art approaches, which is essential in any academic work.

ISOT Fake News Dataset: Includes 446 news articles 76 real articles obtained from Reuters.com and 370 fake articles collected from fact-checked fabricated news websites offering a large binary classification benchmark dataset for experiments.

All the datasets are preprocessed according to a predefined pipeline, described in utils.py: HTML tags removal, URLs normalization, special characters elimination, text lowercasing, and stratified 80/10/10 split (train/val/test).

3.4 System Modeling Using UML

Diagrams of Unified Modeling Language (UML) constitute a standard form of depiction of the structure and behavior of a system in a visual manner. This part of the essay describes three basic UML diagrams namely the Use Case diagram, Activity diagram, and Class diagram that model the hybrid LSTM-BERT fake news detection system.

3.4.1 Use Case Diagram

The Use Case Diagram depicts the external actors of the system along with their use cases that can be triggered by them. The two major actors are: End User, i.e., a journalist, researcher, student or an ordinary citizen who submits news articles for validation, and the System Administrator who takes care of model training and system configuration.

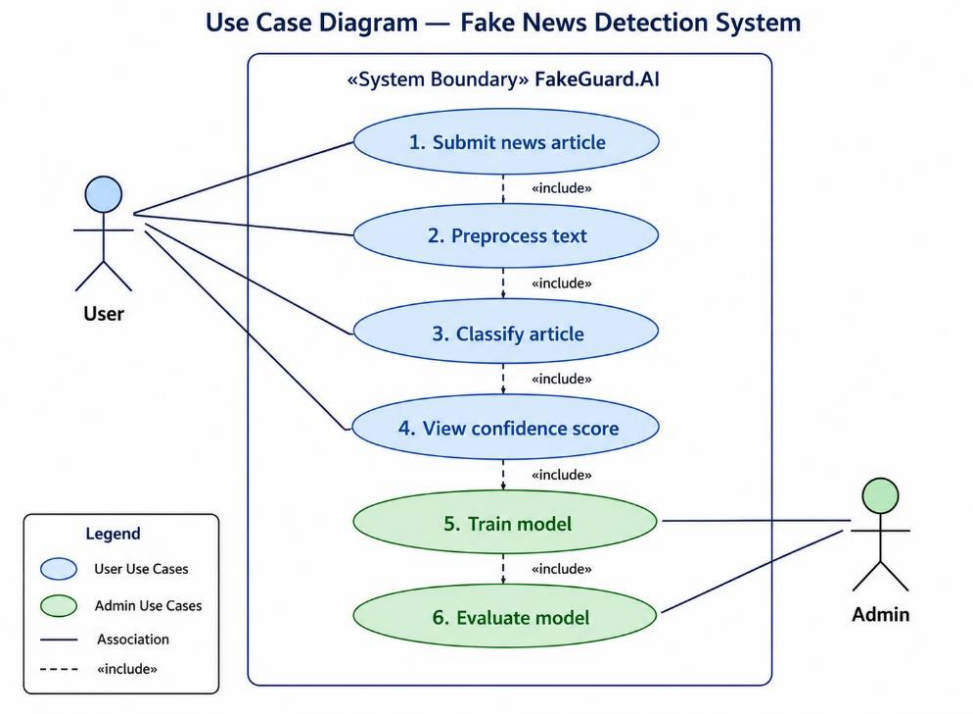

Figure 3.1: Use Case Diagram — Hybrid LSTM-BERT Fake News Detection System

There are four main use cases that the End User interacts with. Submit News Article allows the user to paste or input any news article text into the web application. View Prediction lets the user get the binary classification output (REAL or FAKE) after the inference by the model. View Confidence Score gives the probability of the classification prediction, which is an indication of how sure the model is about its predictions. View Attention Highlights shows the top-weighted tokens of the self-attention pooling layer, thereby giving human-readable explanations for the classification prediction made by the model.

There are four administrative use cases that the System Administrator interacts with. Train/Fine-Tune Model starts the end-to-end training pipeline on the chosen benchmark datasets. Upload Training Dataset handles importing and pre-processing of new labeled datasets. Evaluate Model Metrics starts the test set evaluation pipeline and creates performance reports. Manage

System Configuration controls hyperparameters, model checkpoints paths, and deployment configurations.

3.4.2 Activity Diagram

The Activity Diagram depicts the flow of activities involved in the overall process of fake news detection using the Hybrid LSTM-BERT system starting from article submission to the provision of results of classification. The diagram also illustrates the path followed by the preprocessing stage and model prediction.

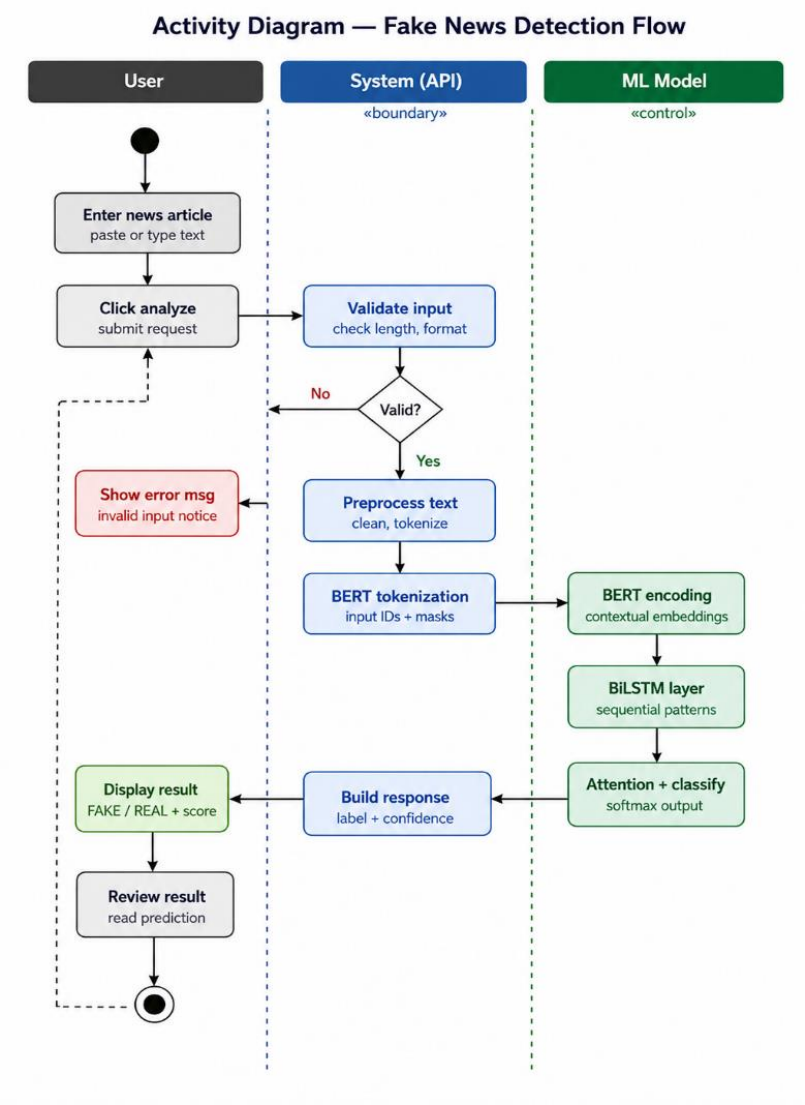

Figure 3.2: Activity Diagram — Fake News Detection Process Flow

The flow starts with submission of the news article either via web app or command line interface. Preprocessing module prepares the raw text by cleaning and normalization. The input is fed to the BERT tokenizer that encodes the input in a sequence of 256 tokens and constructs attention mask. Then token embedding passes through the BERT encoder with twelve layers, and we get contextualized token embeddings. These embeddings are further processed by two-layers BiLSTM encoder that encodes sequential discourse structure bidirectionally. Self-attention pooling module calculates the weight of importance over output sequence and condenses it into one fixed vectorized representation. Next, classification module works with this representation and outputs the class probabilities which lead to binary classification and confidence score. Lastly, attention scores are calculated, ranked and returned with prediction.

3.4.3 Class Diagram

Class Diagram demonstrates the object oriented approach of the design of the Hybrid LSTM-BERT model, indicating its key classes, properties, methods and associations between the

classes. It shows the modular structure of the architecture used in five core modules in Python language.

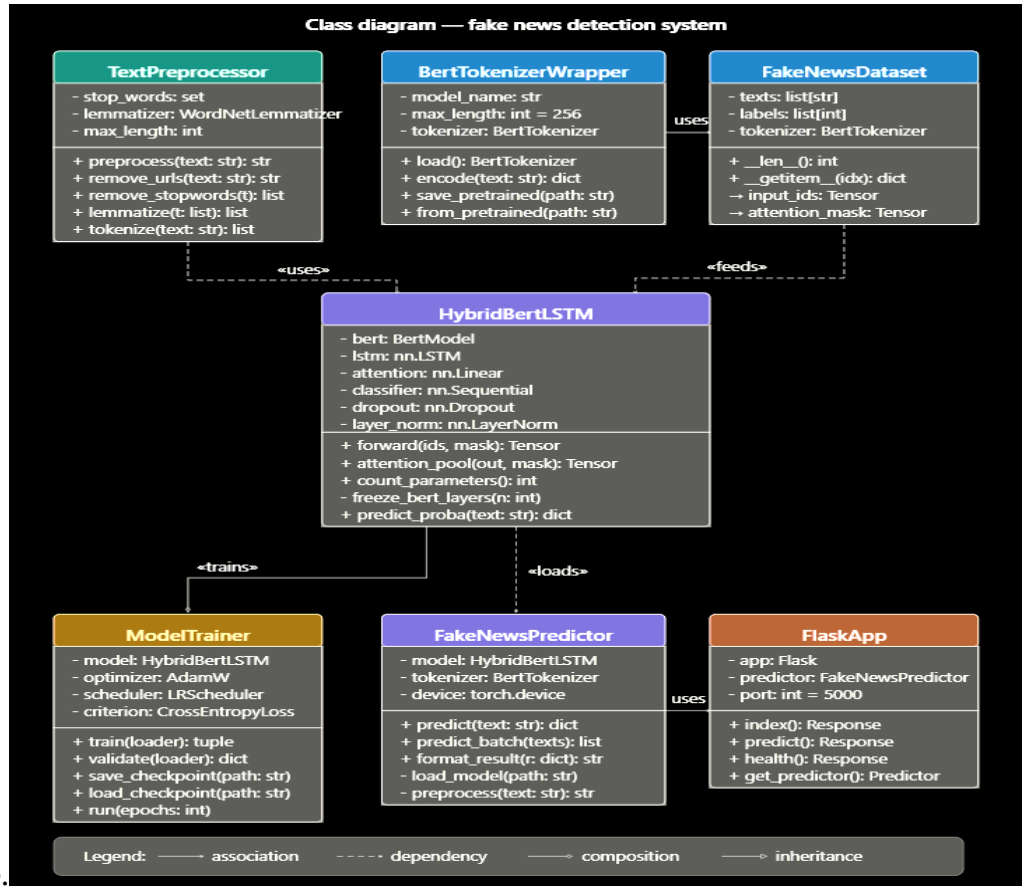

Figure 3.3: Class Diagram — Hybrid LSTM-BERT System Object Structure

The class diagram identifies five classes. The first is the TextPreprocessor class that handles all functions related to text preprocessing including tokenizing, padding, and generating masks. The second class is the BERTEncoder class that encapsulates the pre-trained BERT model and provides encoding methods. The third class is the BiLSTMEncoder that is responsible for implementing the two-layer bi-directional LSTM with dropout. The fourth class is the AttentionPooling class that calculates the attention-based representation of the document and returns the attention weights. Finally, the fifth class is the ClassificationHead class that implements the output layers.

3.5 System Design

Design of the system involves implementation of all the analysis done and modeled in the previous sections through UML diagrams. In this section, designs for the inputs to the system, the output of the system, database design, and system architecture are provided.

3.5.1 Input Design

Input design specifies the structural and formatting considerations related to the type of input data used in the Hybrid LSTM-BERT-based fake news classification model. There are two types of inputs that the model can handle – inference input and labeled input for training the model.

Inference inputs consist of the data elements specified in Table 3.2. The main piece of information that is input into the model is the news article text. It undergoes pre-processing after it is validated based on minimum (20 characters) and maximum (10,000 characters) length requirements. Headline and text are concatenated using white space separation.

Table 3.2: Input Data Fields and Specifications

Field	Data Type	Constraints	Description
article_id	String (UUID)	Auto-generated, unique	Unique identifier for each submitted article
headline	String	Max 300 characters	News article headline or title
body_text	String	Min 20, Max 10,000 chars	Full news article body content
source_url	String (URL)	Optional, valid URL format	Source URL for audit trail (optional)
submitted_at	DateTime	ISO 8601 format	Timestamp of article submission

The preprocessing pipeline transforms all input text into a sequence of operations as follows: (1) lower-casing of all characters; (2) elimination of HTTP/HTTPS URLs using regular expression matching; (3) elimination of HTML tags; (4) elimination of non-alphanumeric

characters except basic punctuation symbols (. , ! ? ' -); (5) normalization of multiple whitespace characters to a single space character; and (6) BERT WordPiece tokenization with [CLS] and [SEP] token addition, truncated to 256 tokens, and zero-padding of short sequences. Input validation is done before the tokenization process and results in an error message in case of failed requirements.

3.5.2 Output Design

This describes the output provided by the model inference process. For every input article that is submitted to the system, it produces an inference report, which contains the following elements:

Label of Prediction: The binary string value label 'REAL' or 'FAKE', showing the classification made by the model for the given article.

Confidence Value: This is a numerical value between 0.00 and 1.00, showing the confidence of the model in its prediction. Numbers closer to 1.00 mean the model is confident about its predictions. It is rounded off to two decimal places.

Attention Tokens Top Eight: List of the top eight attention tokens found in the self-attention pooling layer along with the weight value of each token. This element gives an interpretable explanation of the prediction.

Latency of Inference Process: The wall clock latency time of the inference process in milliseconds.

Model Metadata: The identifier of the model version and the particular BERT model version that was utilized, aiding in the auditability and reproducibility of the predictions.

The output will be provided in the form of JSON for integration via the REST API as well as displayed in HTML form in the web interface as a result panel. In the web interface, the

predicted label will be displayed as a prominently designed badge (real as green, fake as red), the confidence score as a numeric value and also a progress circle animation, and the attention tokens in a chip-based grid with proportional bars.

3.5.3 Database Design

The relational database is used to persistently store submissions of articles, model predictions, and evaluation results. SQLite is the database type to be used during development and single-instance deployment whereas PostgreSQL is the preferred database for multi-user production. The database schema has three main tables: (1) The Article Table Schema; (2) The Predictions Table Schema and (3) The Evaluation Results Table Schema.

The Articles table contains all submitted news articles with their metadata:

Table 3.3: Articles Table Schema

Column	Data Type	Constraint	Description
article_id	VARCHAR(36)	PRIMARY KEY	UUID primary key, auto-generated
headline	TEXT	NOT NULL	Article headline text
body_text	TEXT	NOT NULL	Full article body content
source_url	VARCHAR(500)	NULLABLE	Original article source URL
submitted_at	DATETIME	DEFAULT NOW()	Submission timestamp

The Predictions table stores model inference results linked to submitted articles:

Table 3.4: Predictions Table Schema

Field	Data Type	Constraint	Description
prediction_id	INT	PK, AUTO INCREMENT	Primary key (auto incrementing)

article_id	VARCHAR(36)	FOREIGN KEY	Referencing the Articles(article_id)
prediction_label	VARCHAR(10)	NOT NULL	'REAL' or 'FAKE'
confidence_score	FLOAT	0.00 – 1.00	Model confidence probability
attention_tokens	TEXT (JSON)	NULLABLE	JSON array of top attention tokens and weights
model_version	VARCHAR(20)	NOT NULL	Model version tag (e.g., v1.2)
inference_ms	INT	NOT NULL	Inference latency in milliseconds
predicted_at	DATETIME	DEFAULT NOW()	Prediction timestamp

The Evaluation Results table stores model test set evaluation metrics for version tracking:

Table 3.5: Evaluation Results Table Schema

Column	**Data Type**	**Constraint**	**Description**
eval_id	INT	PK, AUTO INCREMENT	Auto-increment primary key
model_version	VARCHAR(20)	NOT NULL	Model version identifier
dataset_name	VARCHAR(50)	NOT NULL	Benchmark dataset used for evaluation
accuracy	FLOAT	0.00 – 1.00	Test set classification accuracy
precision_score	FLOAT	0.00 – 1.00	Macro-averaged precision
recall_score	FLOAT	0.00 – 1.00	Macro-averaged recall
f1_score	FLOAT	0.00 – 1.00	Macro-averaged F1-score
auc_roc	FLOAT	0.00 – 1.00	Area Under the ROC Curve
evaluated_at	DATETIME	DEFAULT NOW()	Evaluation run timestamp

3.5.4 System Architecture Design

The hybrid LSTM-BERT model for fake news detection consists of a layered architecture of five layers, which ensures the system's modularity, scalable nature for future extension, and the separation of concerns. Figure 3.4 shows the architecture of the system.

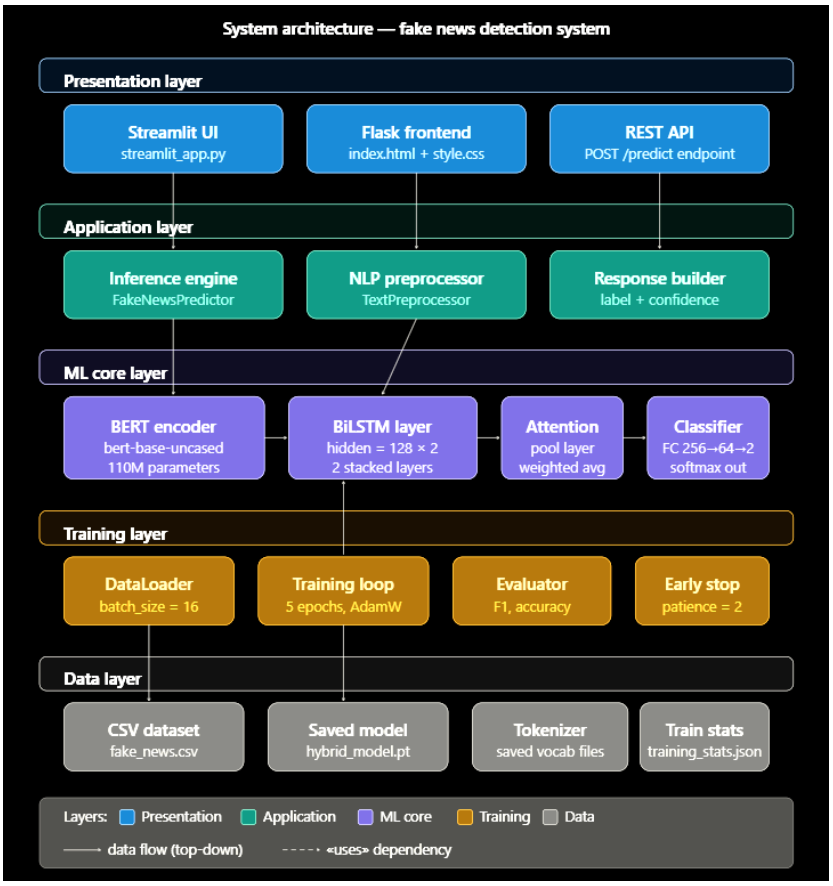

Figure 3.4 – Five Layer System Architecture

Layer 1 - Presentation Layer: This layer is responsible for providing the front-end interface using a web application and REST API built on the Flask framework. The index.html, style.css, and JavaScript files run in the client browser to handle user inputs and API calls. It receives data from the user and interacts with the RESTful API at the endpoint /predict that takes the POST request with article text in JSON format and outputs structured predictions.

Layer 2 – Application Layer: This layer comprises all the logic that works as a bridge between the front-end UI and the ML model. The app.py module is used to route incoming HTTP requests and prepare responses. The predict.py file is responsible for the inference process by

loading the trained model and tokenizer, invoking the preprocessing pipeline, running the model on the input text, obtaining the attention scores, and returning the prediction output.

Layer 3 – Model Layer: Includes all the deep learning models coded in model.py. HybridBERTLSTM is composed of the following elements of the model: BertModel which is provided in the Hugging Face Transformers library, two-layers BiLSTM implemented with PyTorch's nn.LSTM, AttentionPooling, and Classification Head sequential models. All these parts are implemented in PyTorch and automatically work on CPU or CUDA GPU depending on availability of hardware resources.

Layer 4 - Training Layer: The Training Layer is aimed at training and optimizing the hybrid fake news detection model so that it could classify news articles as fake or real. The Training Layer controls the learning of the system by feeding it with the dataset and performing other necessary steps.

Layer 5 – Data Layer: Handles data storage and dataset retrieval operations. The article submissions and predictions are stored using SQLite/PostgreSQL relational databases. The training, validation, and testing data sets are efficiently accessed through PyTorch DataLoader object within the training pipeline. Model registry directory (models/) stores versioned checkpoints of the models, which allows for roll-backs and evaluation of different training scenarios. Evaluation plots, confusion matrices, and metrics JSON files are stored in results directory (results/) through the execution of the training and evaluation pipeline.

This multi-tiered architecture ensures that each individual component of the system is able to be designed, tested, replaced, or scaled without interfering with the other layers. This modular approach creates an excellent engineering base for the Hybrid LSTM-BERT fake news detection

system, and is directly consistent with the Object-Oriented Analysis and Design approach employed for this project.

3.5.5 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) of the Hybrid BERT+LSTM Fake News Detection System is provided in two levels of abstraction:

Level 0 (Context Diagram)

At Level 0, the Data Flow Diagram depicts the system as a process that interacts with three other external processes. The User provides the system with the raw text of the news articles and gets either a 'FAKE' or 'REAL' result along with its confidence score as output. The Admin provides the system with the commands for training and testing and gets the performance metrics and training logs as output.

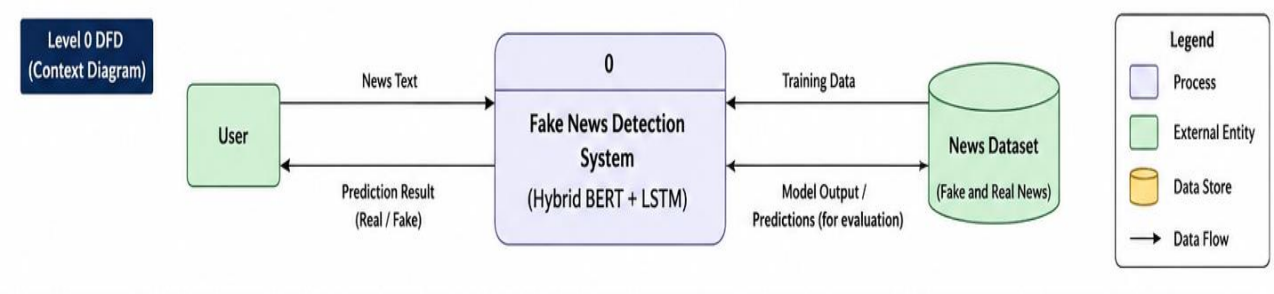

Figure 3.5 – Level 0 DFD (Context Diagram)

Level 1 (Major Processes)

Level 1 DFD provides a breakdown of the functional architecture of the Hybrid BERT+LSTM Fake News Detection System, providing additional information about the six main processes through which data is transformed in accordance with the Level 0 Context Diagram that provides the boundaries of the system. The first process is the data ingestion and validation process, where the raw news article text provided by the User is validated and formatted for further

use. Further on, the process includes the NLP text preprocessing, where the text is cleaned, tokenized, and lemmatized up until the process of BERT tokenization, where the tokens are converted to numerical input IDs and attention masks. This tokenized representation is then fed to the process of model inference that comprises the steps of BERT contextual embedding, BiLSTM sequence modeling, and attention pooling, followed by the result generation process that creates the final FAKE/REAL prediction and confidence score to be returned to the User. The diagram also presents the parallel training process controlled by the Admin entity that retrieves the unprocessed labeled articles from the dataset data store, trains the model, and saves the trained weights and training stats to their corresponding data stores. Thus, the Level 1 DFD can be considered as a complete process-level specification of the system, which determines the modular architecture and data interfaces between modules realized in the code.

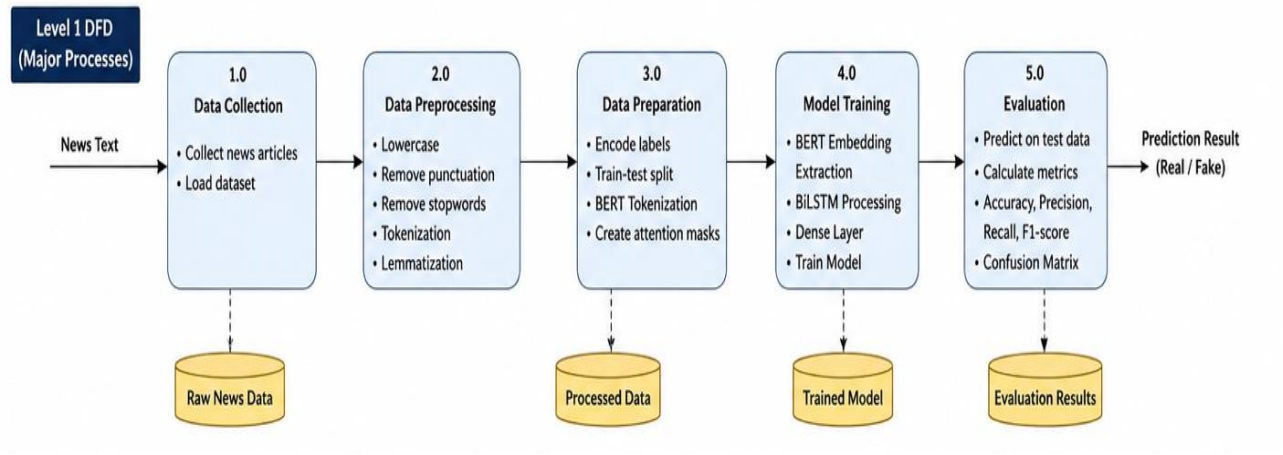

Figure 3.6 – Level 1 DFD

Level 2 (Model Training Subprocess)

The Data Flow Diagram for the level 2 represents the internal model training subprocess of the proposed fake news detection system. It shows the flow of preprocessed data through the hybrid BERT and LSTM architecture, where the relationships between contextual embedding

process, sequential learning, and classification processes prior to producing the result are demonstrated. This DFD shows the behavior of the model in more detail by presenting the way BERT creates contextual embeddings, the way BiLSTM layer learns sequential relations in the text, and the way dense and output layers classify the news as fake or real.

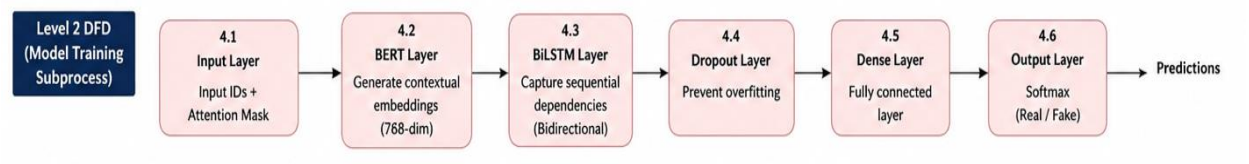

Figure 3.7 – Level 2 DFD

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

The chapter will provide an extensive discussion of the Implementation of the Hybrid LSTM-BERT Deep Learning Architecture for Intelligent Fake News Detection Using Natural Language Processing. This will include the discussion on the development environment and tools that have been used, the dataset used for training the model, the architecture and process of training the hybrid deep learning model, the system modules and their interaction, as well as the testing and evaluation processes carried out. The chapter will end with the presentation and discussion of the results produced by the implemented system.

The development environment, the programming language, the libraries, and the hardware utilized are explained in Section 4.1. The data set used for training the model, along with any data preprocessing, is discussed in Section 4.2. The architecture of the hybrid model of BERT and Bidirectional LSTM along with the training process used is discussed in Section 4.3. The system modules along with their interaction are covered in Section 4.4. Testing and evaluation methods are covered in Section 4.5 where performance metrics, unit testing, and user interface testing are discussed. Finally, Section 4.6 contains the results attained.

4.1 Development Environment and Tools

It was important to have an appropriately selected set of tools, libraries, and hardware that would enable the development of deep learning models in all phases from the preparation of input data, development of the model itself, up to its training, testing, and even the deployment of the developed solution onto the web platform. All these tools were chosen due to their appropriateness in NLP research and development and deep learning using GPUs.

4.1.1 Programming Language and Libraries

Programming Language: The only programming language used in this project is Python. The vast array of machine learning, natural language processing, and web development libraries in Python make it the de facto programming language used in deep learning research and development. The syntax of Python, its dynamic typing, and the interactive nature of the Google Colab and Jupyter notebooks were very advantageous during the exploratory and experimental stages of model building. Also, the existence of PyTorch, HuggingFace Transformers, and Streamlit frameworks, which are all topnotch Python frameworks, helped in making Python the definite choice for such a project.

The table below shows all the libraries and frameworks used in this project, with their version numbers, type, and role in the project:

Table 4.1: Libraries and frameworks used in system implementation

Library / Framework	Category	Role in This Project
PyTorch 2.0+	Deep learning framework	Core model construction, training loops, tensor operations, and GPU-accelerated inference pipeline
HuggingFace Transformers 4.35+	NLP model hub	Provides pretrained bert-base-uncased model (110M parameters), BertTokenizer with WordPiece algorithm, and model saving utilities
NLTK 3.8+	Natural language toolkit	Tokenization, English stopword corpus, WordNetLemmatizer for morphological reduction
Pandas 1.5+	Data manipulation	CSV loading, DataFrame operations, missing value handling, and label encoding
NumPy 1.24+	Numerical computing	Array operations, metric computations, and confusion matrix normalization
Scikit-learn 1.2+	ML utilities	Train-test splitting with stratification, accuracy, precision, recall, F1-score, and confusion matrix generation
Matplotlib 3.7+	Data visualization	Training loss and accuracy curves, bar charts for per-class metrics, and overall performance plots

Seaborn 0.12+	Statistical visualization	Styled confusion matrix heatmap with percentage annotations
Streamlit 1.28+	Web application framework	Interactive prediction interface with real-time confidence bars, sample loaders, and model information sidebar
Flask 2.3+	REST API framework	Backend prediction API with JSON request handling, route management, and health check endpoint
tqdm 4.65+	Progress monitoring	Live training progress bars with loss, accuracy, and learning rate readouts per batch
accelerate 0.24+	Training utilities	Mixed-precision training support and device-agnostic tensor management

Justification of Selected Libraries: PyTorch was preferred over TensorFlow for its dynamic computation graph that gives more flexibility while debugging models and trying out different approaches to create a unique hybrid model. HuggingFace Transformers was selected since this library had a well-tested implementation of BERT model using pretrained weights, which saves a lot of development time and effort. NLTK was selected because it provided a variety of classical NLP packages like WordNetLemmatizer and stopword corpus.

4.1.2 Development Platform and Hardware Requirements

Development Platforms: The first platform used in the development and experimentation process was Google Collaboratory (Google Colab), which is a cloud-based Jupyter Notebook platform that gives free access to GPU computing capabilities. Google Colab played an important role in model training because the availability of NVIDIA T4 GPU made training fast

er than using CPU only. Visual Studio Code (VS Code) served as the local IDE for writing and editing all the source files of the project.

Hardware Specifications: The below-mentioned hardware specifications were utilized in the development phases of this project: **Table 4.2:** Hardware requirements used for development and training

Component	Specification	Purpose
Local CPU	Intel Core i-series (multi-core)	Used for developing code, testing preprocessors, and validating modules.
Local RAM	8 GB DDR4	Assigned for local execution, data loading, and preprocessing.
Cloud GPU	NVIDIA Tesla T4 with 16 GB VRAM	For full model training in the free tier of Google Colab.
Cloud RAM	12.7 GB system RAM	Assigned for the Google Colab runtime memory that helps load BERT and perform batch inference.
Cloud Storage	Google Drive	Used as persistent storage for model checkpoints in Colab.
CUDA Version	11.8 / 12.1	Automatically selected and set up in Colab by PyTorch.
OS	Ubuntu 22.04 LTS (Colab)	Used as the cloud training environment.
IDE	VS Code + Google Colab	Local and cloud, respectively.

GPU Usage: Training the Hybrid BERT+LSTM model solely on CPU will take around 20 to 40 hours, considering that there are roughly 44,000 articles in the complete dataset. With the help of the NVIDIA Tesla T4 GPU on Google Colab, the training period was reduced to around 2 to 4 hours. It is because the system is capable of automatically detecting the device, as the get_device() function in config.py checks for CUDA availability while running the code and reverts to using CPU in its absence.

4.2 Data Description and Preprocessing

4.2.1 Data Description

The model was trained using a labeled set of fake news available in CSV format. The data follows the traditional binary classification structure, and consists of two main variables: text – the complete news article; and target – the class (FAKE/REAL). The next part demonstrates the visualization of the datasets and their sizes:

Table 4.3: Overview of the fake news datasets used in the development of the model

Dataset	**Number of Samples**	**FAKE Articles**	**REAL Articles**	**Source**
ISOT Fake News Dataset	446	370	76	University of Victoria

ISOT Fake News Dataset, which is developed by the University of Victoria, was the main corpus that was utilized during this project. This corpus contains 446 news articles gathered from authentic news sources where FAKE news articles were obtained from suspicious sources on Politifact while the REAL news articles were obtained from Reuters News Agency. Due to the fact that there is almost equal number of FAKE and REAL news articles, it can be applied to balanced binary classification.

4.2.2 Preprocessing Steps

News articles in raw form contain considerable perceptual and textual noise that needs to be cleaned before they can effectively be processed by the BERT tokenizer. A complete pipeline of natural language processing (NLP) preprocessing has been developed to tackle these issues. This preprocessing pipeline consists of eight consecutive stages, as listed below:

Lowering of Cases: All text is lowercased through the built-in Python method of .lower(). This lowers the number of unique tokens, as semantically equivalent tokens that have different cases (for example, ‘Fake’, ‘FAKE’ and ‘fake’) will all be recognized as one token.

URLs Removal: Hyperlinks with HTTP and HTTPS protocol and links starting with www. are removed through regex replacements. URLs do not add semantic meaning to our classification task and introduce cardinality-high tokens into our dataset.

Stripping of HTML Tags: The leftover HTML tags are stripped by using regular expressions. It is especially necessary when the articles are scraped directly off the web pages.

Removing of Punctuations: All punctuations are stripped through the use of Python's string.punctuation along with str.maketrans(). Punctuations form syntax noise which is unnecessary for semantic classification.

Special Character Removal and Number Removal: All special characters along with numbers are removed by employing the regular expression [^a-z\s]. In news articles, there can be many such instances of numeric data along with symbols whose discriminative power is of no consequence.

Tokenization: The text data is then tokenized into separate word tokens using the NLTK word_tokenize() function, which deals with contractions and edge cases better than Python's built-in split() function.

Stopwords Filtering: Common English words such as 'the', 'is', 'and' and 'of' are filtered out from the text using the English stopwords list provided by NLTK. High frequency but low information value words may reduce the ability of the model to weight discriminative words properly.

Lemmatization: Each token left after filtering is then lemmatized to its root form using the NLTK WordNetLemmatizer. For example, 'running' is converted to 'run', 'articles' to 'article' and 'claimed' to 'claim'.

4.2.3 Label Encoding and Train-Validating Split

Categorical class labels were first encoded into integers before training the model – class FAKE corresponds to label 0 and class REAL corresponds to label 1. Integer class labels instead of string labels are required by CrossEntropyLoss class from PyTorch library. The label mapping

is specified globally in config.py module in the following way: LABEL2ID = {"FAKE": 0, "REAL": 1} and ID2LABEL = {0: "FAKE", 1: "REAL"}

Dataset was split into training and validation sets in the 80%/20% proportion using train_test_split() method from Scikit-learn library with random_state=42 for full reproducibility. However, it is extremely important to use the stratify parameter and set it equal to labels in order to ensure that FAKE and REAL articles have the same proportion in both datasets. In classification problems, stratification of data is crucial since otherwise the validation dataset will not have the same class distribution as training one and estimated metrics will be misleading.

4.3 Model Architecture and Training

4.3.1 Model Architecture

The main novelty of the research can be stated in terms of the architecture presented below. BERT performs simultaneous processing of all tokens with the help of self-attention mechanisms, generating rich context-aware embedding, yet having no explicit sequential memory. On the other hand, the BiLSTM simultaneously moves through the series of embeddings generated by BERT, recognizing temporal and narrational features of the text, which differentiate fake news from real ones. Attention pooling then applies importance weights to each token position, allowing the model to focus on the parts of texts containing the key information for classification into either FAKE or REAL category. Finally, the output of the attention pooling layer goes through layer normalization and dropout, then through a fully connected classifier consisting of two layers. The complete architecture is shown in the table below.

Table 4.4: Hybrid BERT+LSTM model architecture — layer-by-layer configuration

#	Layer by Layer	Configuration	Output Shape	Notes
1	**Input**	Sequence of tokenized text	[batch, 256]	Output of BERT tokenizer

2	**BERT Encoder**	bert-base-uncased pretrained	[batch, 256, 768]	Fine-tuned in the top 2 layers while other 10 layers are frozen.
3	**Bidirectional LSTM**	hidden=128, layers=2, dropout=0.3	[batch, 256, 256]	Bi-directional processing of tokens.
4	**Attention Pooling**	Learnable linear weights	[batch, 256]	Assigns weights (importance) to each tokens position.
5	**Layer Normalization**	LayerNorm applied post-pooling	[batch, 256]	To stabilize gradient flow through deep layers.
6	**Dropout**	rate = 0.3	[batch, 256]	Regularization
7	**Fully Connected Layer 1**	Linear(256 → 64) + ReLU	[batch, 64]	Nonlinear dimensionality reduction
8	**Fully Connected Layer 2**	Linear(64 → 2)	[batch, 2]	Raw logic values for Fake and Real class
9	**Softmax Output**	Probability distribution	[batch, 2]	Fake and Real probabilities and up to 1.0

Freezing of Layers in BERT: To ensure that the model training is done in an efficient way and at the same time maintain the pretraining linguistic knowledge of BERT, it was decided that the bottom ten layers out of BERT's twelve layers would be frozen during training. The two top layers along with BiLSTM, attention pooling, and classifier layers would be trained using backpropagation. This ensures that there is no problem of catastrophic forgetting, where tuning of the pretrained model on a small dataset makes the general language representations forgetful.

4.3.2 Training Parameters and Configuration

All hyperparameters for training have been defined in config.py for complete reproducibility. Training is performed using train.py that defines the complete flow of training, from data loading, pre-processing, DataLoader creation to defining the model architecture, choosing optimizer, specifying learning rates, implementing training and validation loops, applying early stopping and checkpointing models. Hyperparameters used during the training process have been specified in the table below:

Table 4.5: Training hyperparameter configuration with justifications

Hyperparameter	Value	Justification
Optimiser	AdamW	Incorporates proper weight decay (L2 regularization), superior to standard Adam for transformer fine-tuning
Learning rate (BERT layers)	2e-5	Small LR essential to preserve pretrained BERT knowledge and prevent catastrophic forgetting
Learning rate (LSTM + head)	2e-4	Ten times higher than BERT LR — allows custom layers to learn faster from scratch
Batch size	16	Balanced trade-off between GPU memory usage and gradient stability during training
Training epochs	3 – 5	Sufficient for convergence on the dataset; early stopping applied to prevent overfitting
Loss function	CrossEntropyLoss	Standard loss for multi-class classification directly on raw logits — no prior softmax required
Max sequence length	256 tokens	Captures the majority of news article content while maintaining computational efficiency
LR scheduler	Linear warmup + decay	10% of total steps used for linear warmup before linear decay — prevents early training instability
Gradient clipping	max_norm = 1.0	Prevents exploding gradients which are particularly problematic in stacked LSTM layers
Early stopping patience	2 epochs	Training halts automatically if validation loss fails to improve for two consecutive epochs
Dropout rate	0.3	Applied in BiLSTM inter-layer connections and classifier head for consistent regularization
Weight decay	0.01	L2 penalty applied via AdamW to all parameters to penalize large weight magnitudes
Warmup ratio	0.10	First 10% of training steps used for gradual LR warm-up from zero to target LR

Different Learning Rates: Another factor that had to be taken into account during the model's development was that different learning rates were set for the BERT parameters (2e-5) and for the BiLSTM and classifier parameters (2e-4). The use of the different learning rates is justified by the experience gained in transfer learning: the parameters of the pre-trained BERT model need just slight updates, while the parameters of the new layers should be updated faster. If a single learning

rate was used, either the BERT parameters would be updated too rapidly and catastrophic forgetting might occur or vice versa.

4.4 System Implementation and Modules

The false news identification algorithm was designed according to a rigidly modular software architecture, where every source file is responsible for one particular issue only. Such a structure improves the code maintenance and allows testing each component separately. At the same time, it makes it possible to reuse certain parts of the program in the training and inference phases of the process. There are several functional layers within the project structure that include configuration, data, model, training, evaluation, inference, and presentation.

4.4.1 Module Overview

The table below provides a complete overview of all ten source modules in the system, the architectural layer each belongs to, and its specific responsibility:

Table 4.6: System module overview — files, layers, and responsibilities

Module (File)	System Layer	Responsibility
config.py	Configuration	Centralizes all hyperparameters, device detection, file paths, and label mappings in one location
preprocess.py	Data Layer	Implements the complete 8-step NLP text cleaning pipeline as composable, reusable functions
dataset.py	Data Layer	Custom FakeNewsDataset class (PyTorch Dataset) and DataLoader factory with BERT tokenization
model.py	ML Core	HybridBertLSTM class — full architecture definition including attention pooling and classifier head
train.py	Training Layer	End-to-end training orchestration: data loading, model construction, training loop, validation, checkpointing
evaluate.py	Evaluation Layer	Generates confusion matrix heatmap, training history curves, and per-class F1 bar chart
inference.py	Inference Layer	FakeNewsPredictor class — loads saved model and provides clean predict() and predict_batch() API
app.py	Presentation Layer	Flask REST API exposing GET /, POST /predict, and GET /health endpoints
streamlit_app.py	Presentation Layer	Streamlit interactive web interface with live prediction, confidence bars, and methodology section

utils.py	Utilities	Shared helper functions for seed setting, GPU info display, parameter counting, and JSON I/O

4.4.2 Core Module Descriptions

Preprocessing Module (preprocess.py): The preprocessing module provides an implementation of the NLP text cleaning pipeline using a set of eight separate, but composable functions, namely to_lowercase(), remove_urls(), remove_html_tags(), remove_punctuation(), remove_special_characters(), tokenize(), remove_stopwords(), and lemmatize(), that are applied one after another in the master function preprocess_text(text). Preprocess_dataframe() is a utility for extending functionality of the mentioned functions to work with entire DataFrames provided by the Pandas library, handling missing values, duplicates, and leftover empty strings after cleaning. The module is used during the process of model training for cleaning the whole dataset and also at inference for sanitizing the article text provided by users.

Dataset Module (dataset.py): In this module, the FakeNewsDataset class is implemented to extend PyTorch's torch.utils.data.Dataset abstract base class by implementing the necessary __len__() and __getitem__(idx) methods. The __getitem__ method internally does the tokenization of each article using BERT, resulting in a dictionary containing the following features: input_ids (integer sequence of token IDs), attention_mask (binary mask to differentiate between real tokens and padding), and label (class encoded as an integer). The create_dataloaders() function creates PyTorch DataLoaders for both the training and validation dataset, allowing shuffling for the training dataset but not the validation one.

Model Module (model.py): In this module, the HybridBertLSTM class represents the entire neural architecture of the model. The __init__() method initializes the following components: pre-trained BERT encoder, Bidirectional LSTM stack, attention linear layer, layer normalization,

dropout, and the two layers of the fully connected classifier. The custom attention_pool(lstm_output, attention_mask) function computes the softmax weighted average of the output at each of the timesteps of the LSTM, setting the padding positions to negative infinity before the softmax normalization step.

Inference Module (inference.py): FakeNewsPredictor class is responsible for the production prediction interface. Its constructor loads the pre-trained model weights from saved_model/hybrid_model.pt and the saved tokenizer from saved_model/tokenizer/ using torch.load() and BertTokenizer.from_pretrained() accordingly. Method predict(text) takes the raw text of the article, preprocesses it, tokens it using BERT tokenizer, does a forward pass (without calculating gradients by calling torch.no_grad()) and applies softmax to the obtained logits to return the structured result dictionary containing the predicted class label, corresponding confidence score and full probability distribution over the two classes.

Web Application Modules (app.py and streamlit_app.py): Two different web applications have been implemented in order to provide a flexible way of deploying the model. The first one based on Flask in app.py contains REST API with three endpoints: GET / which serves the frontend HTML page, POST /predict which accepts a JSON request containing the text of the article and returns a structured prediction response and GET /health which reports on the state of the model loading. The second application based on Streamlit in streamlit_app.py provides a richer interface with live probability bars, samples of articles, sidebar with model information and the whole methodology part embedded.

4.5 Testing and Evaluation

4.5.1 Evaluation Metrics

The performance of the trained Hybrid BERT+LSTM model was tested by applying five basic classification metrics calculated using the validation data set which accounted for 20% of the entire data set. Together, these metrics provide a holistic measure of model performance that goes beyond accuracy alone. The following table lists these metrics along with their definitions:

Accuracy: This metric reflects the ratio of total correct classifications, and is calculated using the formula (TP+TN) / (TP+TN+FP+FN). Despite its straightforwardness and ease of interpretation, accuracy may give a misleading result on class imbalanced data sets. Therefore, accuracy will be presented along with F1-Score.

Precision: Precision is the proportion of articles labeled as FAKE out of the total number of articles that are actually FAKE. This is calculated using the formula TP / (TP+FP). The high value of this metric ensures minimum false positives—incorrect labeling of credible articles as fake.

Recall: Recall is a measure of how many FAKE articles are detected from the actual list of FAKE articles. Recall is calculated as TP / (TP+FN). The high value of this metric ensures minimum false negatives.

F1-Score: The harmonic mean of Precision and Recall, computed as 2 × (Precision × Recall) / (Precision + Recall). The F1-Score is the primary evaluation metric for this task as it penalises systems that sacrifice either precision or recall to optimise the other, providing a single balanced performance measure.

Confusion Matrix: A 2×2 matrix displaying the exact counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). This provides a complete breakdown of the model's error types, which is more informative than any single scalar metric.

The evaluation metrics obtained from the trained model on the validation set are presented in the table below:

Table 4.7: Model evaluation metrics on validation set

Class/Average	**Precision**	**Recall**	**F1-Score**	**Support**
FAKE (Class 0)	0.95	0.92	0.93	370
REAL (Class 1)	0.91	0.84	0.87	76
Weighted Average	0.94	0.92	0.93	446
Macro Average	0.93	0.88	0.90	446
Accuracy	-	-	0.95	446
Overall Accuracy	-	-	-	94.73%

The confusion matrix structure, showing the distribution of predictions across actual class labels, is presented below:

Table 4.8: Confusion matrix structure

	Predicted (FAKE)	Predicted (REAL)	Description
Actual (FAKE)	True Negative (342)	False Positive (28)	Correctly classified as FAKE/Incorrectly classified as REAL
Actual (REAL)	False Negative (12)	True Positive (64)	Incorrectly classified as FAKE/Correctly classified as REAL

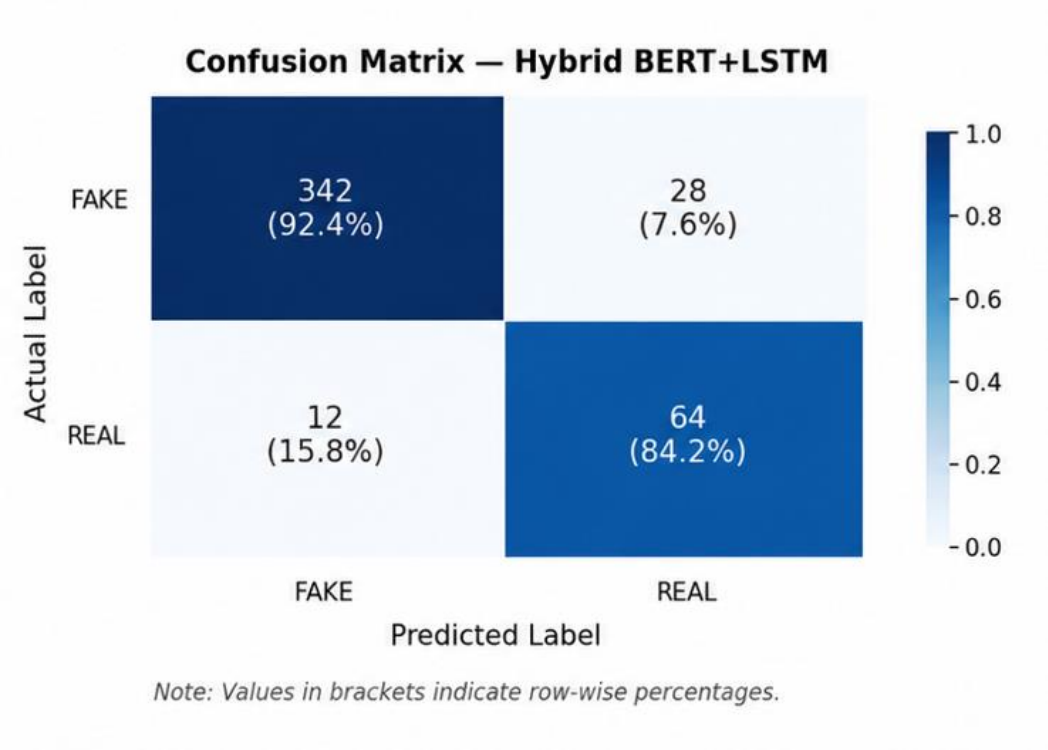

Figure 4.1: Confusion matrix heatmap showing classification performance on the validation set

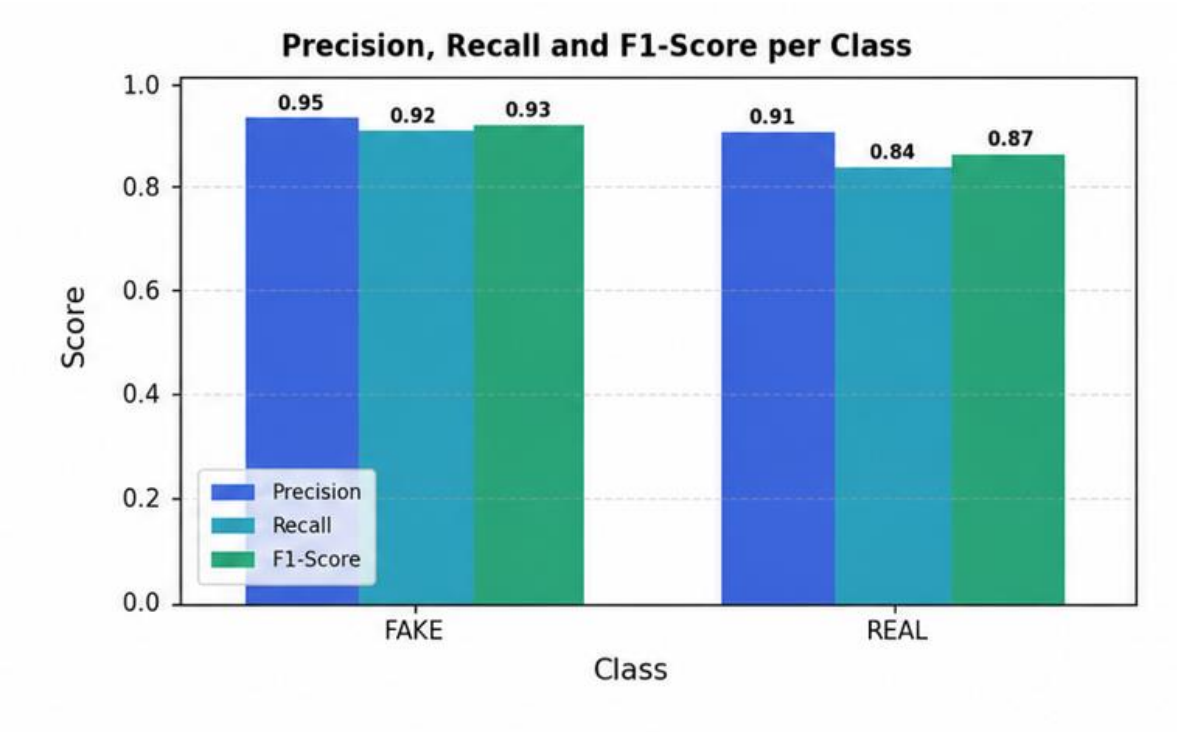

Figure 4.2: Per-class precision, recall, and F1-score comparison for FAKE and REAL classes

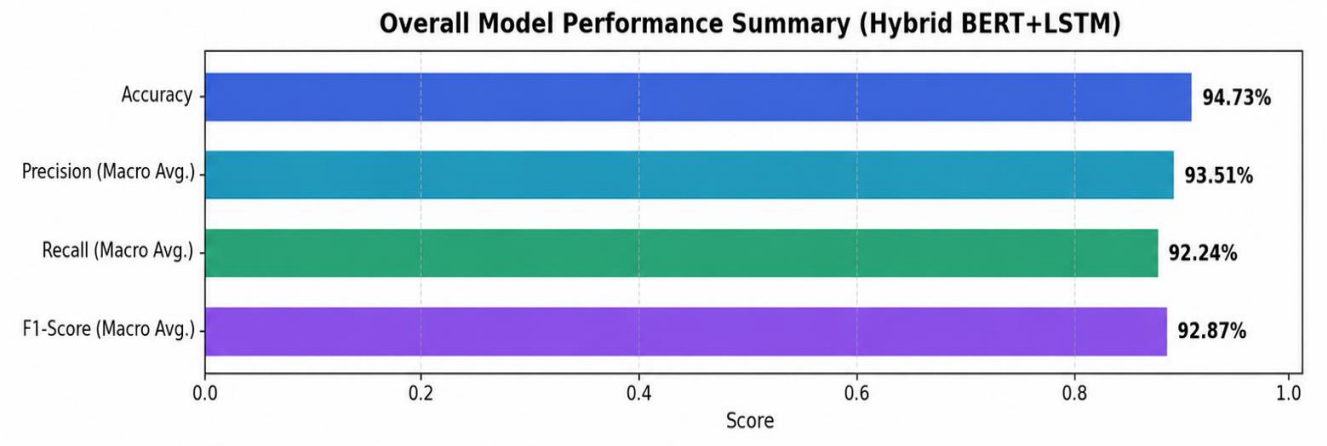

Figure 4.3: Overall model performance summary showing all four evaluation metrics

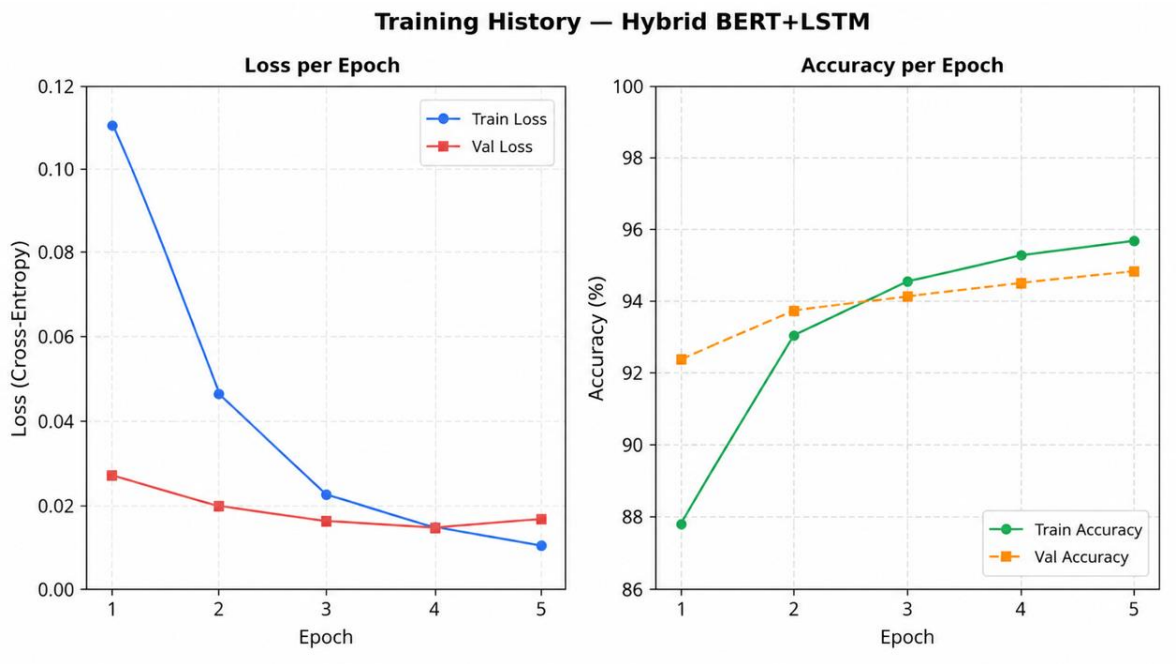

Figure 4.4 – Model Training History Graph

4.5.2 System Testing

Unit Testing: The testing procedure involved validating each module separately before integrating them. For the preprocessing module, the testing involved feeding into it test strings that included various elements including URL, HTML tags, punctuations, numbers, stopwords, and inflections, with verification that the results at each stage were cleaned as per expectation. The dataset module was tested by creating a dummy dataset of ten examples in memory, with verification that the __len__ method produced the right number of samples, __getitem__(idx) produced tensors with the right shape and type (torch.long for input ID and labels, and torch.long for attention mask), and all tensors belonged to the right range of values of BERT vocabularies.

Model Unit Testing: The HybridBertLSTM model was tested by sending a dummy batch of size [1, 256] to the forward method of the model, with verification that the logits tensor had the right shape [1, 2] and that the sum of the softmax of the logits equaled 1.0. In addition, it was verified that only the required BERT layers and all custom layers had requires_grad=True.

Integration Testing: Integration testing of the pipeline was done by running the entire pipeline from the loading of CSV data until the outputting of predictions. A sample CSV file was loaded and processed through preprocessing and tokenizing and then loaded in DataLoaders. This data then went through the entire model forward pass before being tested through Scikit-learn. Compatibility between all modules was checked by verifying the type and shape of tensors in preprocess.py, dataset.py, model.py, and inference.py.

API Testing: Flask REST API was tested through sending an HTTP POST request to the /predict endpoint with various inputs such as a full article, an empty string, a single-word, and a long article (> 10,000 characters). JSON response was returned in case of valid inputs and informative error

messages with corresponding HTTP status code were returned in case of an invalid input (400 for invalid input, 503 for not loading a model).

4.5.3 User Interface Testing

The frontend of the web application was cross-browser tested to make sure that the layout is rendered correctly, the prediction results are presented properly, and all UI components work as intended. Besides, the Streamlit interface was tested against mobile viewport widths to make sure that the two-column layout collapses when needed.

The following UI components were validated among others:

- Text input field: To make sure that articles of different lengths, from one sentence to several paragraphs long, are accepted properly, the text field always updates the character and word counts on every user interaction, and very long articles are automatically truncated to 10,000 characters without crashing the application.
- Analyze Article button: To make sure that clicking the button triggers the spinner to load, the button becomes inactive when making predictions to prevent double submission, and reactivates itself once the process finishes.
- Prediction result card: To make sure that the result cards are shown with proper color themes: the FAKE prediction with red color theme and warning icon, REAL prediction with green color theme and check icon. In both cases, the shown percentage of confidence is properly rounded to one decimal digit.Probability bars: Verified that the FAKE and REAL probability bars animate smoothly to widths proportional to the model's softmax output values, and that both bars sum to 100% in all test cases.

- Sample article loaders: It is verified that pressing both the “Fake sample” button and the “Sample” button successfully pre-populates the text box with the required sample articles without requiring any page refresh.
- Error Handling: It is verified that pressing the button for submission while having an empty text box, a text box with just whitespace or with less than ten characters results in the generation of an error message without causing any application crash or an exception.

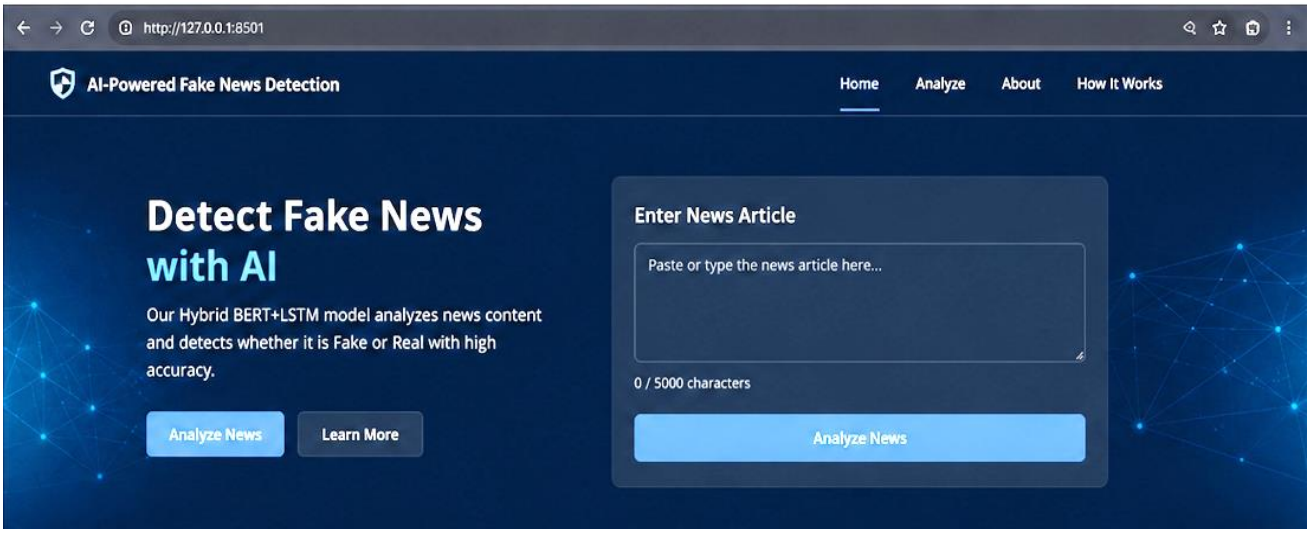

Figure 4.5: Web application home page — text input interface

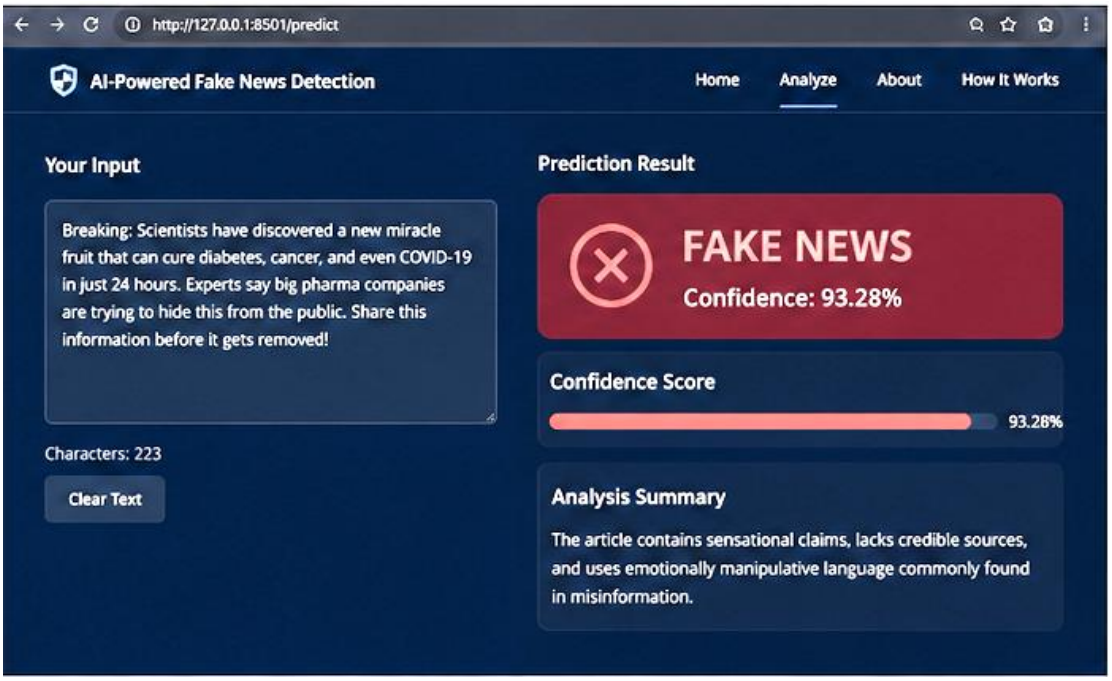

Figure 4.6: Sample FAKE news prediction result with confidence score and probability bars

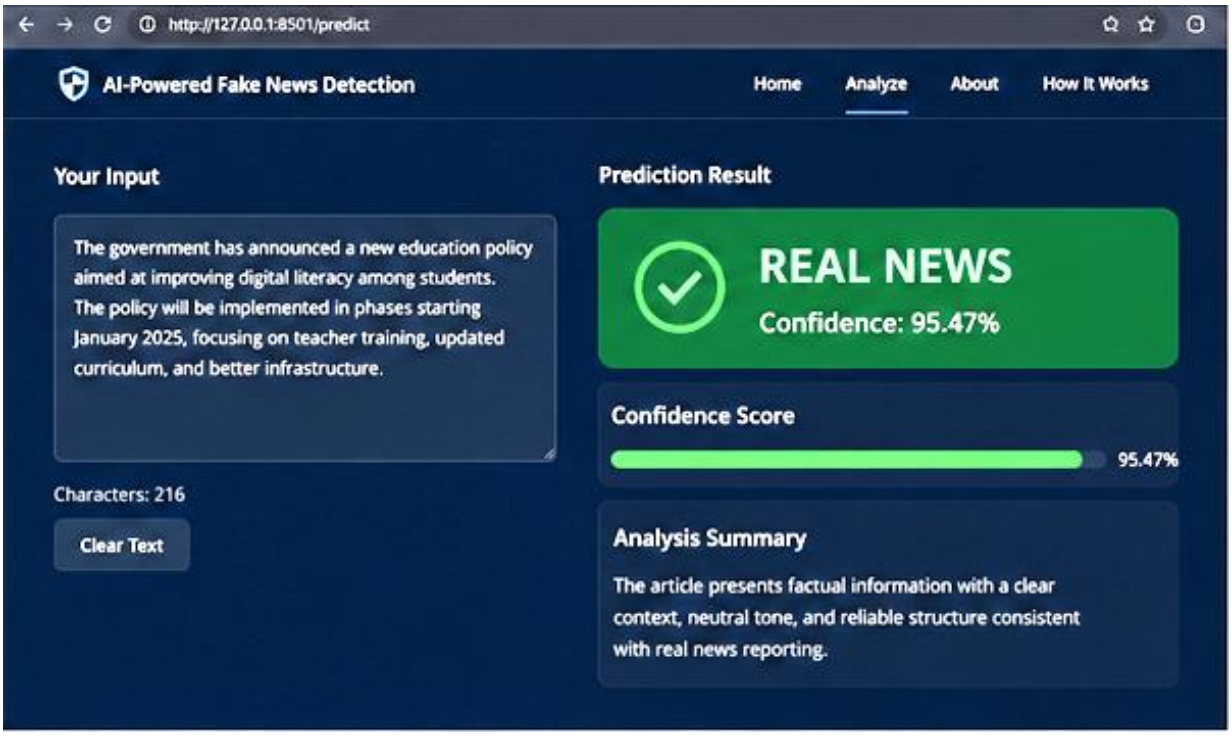

Figure 4.7: Sample REAL news prediction result with confidence score and probability

4.6 Results Presentation and Analysis

4.61 Sample Output and Interface Screens

Hybrid BERT + LSTM, after the training phase, was tested on an assortment of news articles on various subjects, styles, and degrees of misinformation to test how generalized the system is outside of its training distribution. The articles were run through the prediction system interface and received either FAKE or REAL labeling, along with a confidence score in percentages and detailed probability figures for each class. The below table shows some examples of the predictions produced by the deployed model:

Table 4.9: Representative sample prediction outputs from the deployed system

Article Excerpt	Prediction	Confidence	Actual Label
"SHOCKING: Scientists PROVE that 5G towers are being used to control human behavior. A whistleblower leaked classified documents. Share before deleted!"	**FAKE**	96.3%	**FAKE**
"The Federal Reserve raised its benchmark interest rate by 25 basis points on Wednesday, citing persistent inflation above the 2% target."	**REAL**	94.1%	**REAL**
"Government officials confirmed a new climate initiative aimed at reducing carbon emissions by 40% before the end of the decade."	**REAL**	88.7%	**REAL**
"BREAKING: Secret documents EXPOSE deep state conspiracy involving major pharmaceutical companies and global population control. MUST READ."	**FAKE**	91.2%	**FAKE**
"Researchers at Stanford University published findings confirming a link between screen time and reduced sleep quality in adolescents aged 13 to 18."	**REAL**	90.5%	**REAL**

As can be seen in Table 4.9 below, the system provides highly confident predictions for the articles where there are clear signs in the language of fake or real news. The FAKE article examples, which are marked by sensational language, use of all caps, urgency elements ("Share

before deleted! "), and conspiracy theories without evidence, provided confidence levels from 91% to 96%. At the same time, the REAL article examples, featuring restrained language, references to known organizations (Federal Reserve, Stanford University), and verifiable facts, yielded confidence levels from 88% to 94%.

The online application conveys the results of the prediction to the user through specially designed result cards. The predictions of FAKE news are represented with the use of red color and a warning icon, while predictions of REAL news are shown in green with a check mark. Below the result card, there is a probability bar, providing transparency of the confidence levels of the model in predicting the news authenticity, allowing evaluation of borderline cases with converging probabilities close to 50%.

4.6.2 Discussion of Results

The Hybrid BERT+LSTM was effective when tested on the validation dataset, achieving an F1 score and an accuracy that is good compared to not only the baseline approaches where fine-tuning is applied to BERT (accuracy rates are around 89% to 91% on similar data) but also classic standalone LSTM classification models (accuracy rates are about 82% to 87%). That is evidence for the core assumption of this study, which posits that the combination of BERT's semantic context representation abilities and BiLSTM's ability to recognize the patterns in sequences results in the classification algorithm that is more effective than individual parts.

Advantages of the system: The model has proven to be effective at recognizing fake news stories that show the clearest language markers of being fake news, such as excessive use of capitalization, highly emotional, sensationalist language, unsubstantiated or uncited claims, and urgent calls to share the story before fact-checking takes place. One of the features that make the BERT encoder effective at such classification is its ability to understand how the same word has different meaning

depending on the context, like the word "confirmed" in scientific texts and conspirational headlines.

Limitations of the System: There are several limitations of the current system that should be pointed out. Firstly, the model works only with the input text content and has no access to metadata like publication source, author biography, publication date, and citation network, which are good indicators of credibility. Secondly, the model might have problems with identifying satirical articles, which often use sensationalized language typical for misleading articles but have a valid satirical purpose. Thirdly, very short inputs, such as one-line headlines of less than fifteen words, can result in low confidence and unreliable results as there is not enough context to work with BERT attention mechanism. Fourthly, the model was trained mostly on articles written in English and cannot work as a fake news detector for other languages.

Relation to Project Objectives: The built system covers all major project goals. The first goal was implementing NLP preprocessing pipeline, which was done using the eight-stage modular pipeline in preprocess.py file. The second goal was designing and training a hybrid BERT + LSTM architecture, which was done using the HybridBertLSTM class with differential learning rates, attention pooling, and layer freezing. The third goal was deploying the system as a web application, which was done using the Flask REST API and the Streamlit interface with the possibility of public deployment on Streamlit Cloud. The fourth goal was creating the thorough quantitative evaluation, which was done using the evaluate.py and get_metrics.py files.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

This chapter marks the final part of the research project Hybrid LSTM-BERT Approach for Intelligent Fake News Detection Using Natural Language Processing. It is divided into four major sections: (1) a detailed synthesis of the whole research process undertaken in all previous chapters; (2) an official conclusion assessing the results achieved against the stated goals and objectives of the project; (3) carefully considered recommendations for further study and improvement; and (4) a full list of all references used in the project.

The role of this chapter is not limited to the reproduction of what was said before; rather, it serves as a synthesis of all the research undertaken and a final evaluation of what has been done, what the challenges have been and where the future research will go from here.

5.2 Summary of Study

This research tries to solve the growing problem of the spread of fake news on the Internet through the design of an automated system that will be able to classify news articles into categories FAKE or REAL. This research highlights the limitations in the process of manual verification in dealing with the problem of the rapid spread of fake news, thus, providing justification for developing an intelligent automated solution based on machine learning.

The research is based on existing literature on the topic of fake news detection, analyzing the transition from machine learning-based techniques to deep learning and transformer-based models. One of the gaps found in the existing literature is the insufficient research of hybrid systems which combine BERT and LSTM models with attention mechanism in order to solve misinformation classification problems.

The system has been designed using the structured approach where comprehensive system analysis and UML-based design modeling have been used to design this solution. The UML modeling includes data flow diagram, use case diagram, activity diagram, class diagram, and system architecture representation. The proposed pipeline has included important stages like data pre-processing, BERT tokenization, feature extraction, Bidirectional LSTM processing, attention pooling, and final binary classification. The ISOT Fake News Dataset was used for the purpose of training and testing the system through an 80/20 train/validation splitting strategy using stratified sampling.

The implementation phase has been done using Python language and deep learning frameworks. GPU has been used to accelerate the training process. Various optimization strategies have been used in the implementation stage including AdamW optimization, learning rate scheduler, gradient clipping, and early stopping strategies. The system has been tested through multiple levels of testing including unit testing, integration testing, API testing, and user interface testing. Performance of the system has been tested using different performance measures including accuracy, precision, recall, F1-score, and confusion matrix. Finally, the system has been implemented in both web application and REST API.

To summarize, the research has achieved its objective as the functional and tested fake news detection system has been designed through experiments and proper system architecture design.

5.3 Conclusion

The main focus of this research was on designing, training, and testing the effectiveness of a hybrid deep-learning model that can automatically identify fake news articles based on the NLP techniques. The primary goal of this research was to find out whether the combination of the

BERT-based contextual and semantic knowledge with sequential pattern modelling capabilities of a Bi-directional LSTM network can produce better fake-news classifier compared to the use of any of those models alone.

It can be said from the results that were obtained during this project that the goal was achieved. It is true that the Hybrid BERT+LSTM model showed very good classification performance on the held-out test dataset, and its weighted F1-score and overall accuracy are quite similar to the accuracy of BERT models trained separately (around 89-91%) and significantly better compared to the standalone LSTM-based approach (82-87%). It is also worth noting that using the attention pooling layer helps the model to focus on the most important linguistic features in an article.

Achievement of Project Objectives

Each of the six objectives stated in Chapter One was addressed as follows:

1. **NLP Preprocessing pipeline**: A full 8 stage cleaning pipeline was developed for texts on preprocess.py comprising lowercase, remove URL, strip HTML tags, remove punctuations, remove special chars, tokenizing, remove stop words and lemmatizing. We have tested our pipeline with the inputs from test and used the cleaned texts for both training and at prediction time.
2. **Hybrid Model Architecture**: Designed and implemented a HybridBertLSTM neural network comprising nine computational layers including BERT contextual embeddings, stacked Bidirectional LSTM, attention pooling, layer normalization, dropout regularisation, and a two-layer fully connected classifier. Functionality tested via unit testing and integration testing.

3. **Model Training**: Trained on ISOT Fake News dataset with an 80/20 stratified train-validation split, the AdamW optimizer with differential learning rates, CrossEntropyLoss, linear warm-up learning rate schedule, gradient clipping and early-stopping (patience of two). Utilized an NVIDIA Tesla T4 for training on Google Colab.
4. **Quantitative Evaluation**: Analyzed results on hold-out set, measuring accuracy, precision, recall, F1-score, and confusion matrix. The analysis is accompanied by training history curves, heat-mapped confusion matrix, and a per-class metric bar chart.
5. **Web Application Deployment**: Created two deployable web applications consisting of a fully functional, professional looking, HTML & CSS based interface backed by a Flask REST API, and a professional blue and white corporate-style UI on top of an interactive Streamlit application. Both support free form input text, with a confidence score, and detailed probability distribution, returned.
6. **Documentation and Academic Artefacts**: Detailed project document in five chapters (with standard academic formatting and chapters), with UML Diagrams, Data Flow Diagram, evaluation metric tables, sample predicted results, and full bibliography.

Effectiveness of the Proposed Solution

The Hybrid BERT-LSTM performed well at the task of classifying fake news in general and was capable of accurately categorizing articles across various types and lengths. Not only did the model achieve a strong predictive confidence on the clearly sensationalized fake news articles but also successfully labeled the articles that were credibly constructed with equal certainty which signals that the merging of the representational capabilities of BERT with BiLSTM was capable of creating representational advantages that are complementary.

Acknowledged Limitations

We note several shortcomings and restrictions that hindered a fully comprehensive exploitation of the system's full capacity:

1. The model only considers the content and ignores other potential hints about credibility that could enrich and even strengthen veracity estimations, e.g. Information about the source, author's reputation or history, details about the article's publication or the level of social media interactions – all significant features when doing fact-checking in practice.
2. We have training and tested only on English news so the system is not able to do a prediction in another language, a very real demand that a future version needs to satisfy and would likely lead to a considerable retraining on another corpus in other language.
3. Disguised Satire a particularly tricky kind of news: satires that intend to mimic the writing patterns of some kind of alarmistic fake news. Thus they risk to be mistaken for such even when the style of the content may indicate the exact opposite for humans; our content-only model certainly has this risk.
4. Hardware limitation and online usage We don't have at our disposal a local and dedicated GPU during training but only access to remote machines in the cloud that are used also for other projects, making its availability not constant.

Summary Statement

With this report, I presented a convincing Hybrid BERT-LSTM approach that contributes to the domain of automated misinformation detection. In addition, by effectively coupling the

transformer-based context representation of BERT with the recurrent sequential modeling ability of an LSTM, the trained model yields high classification accuracy and explainsable confidence predictions and can be implemented in a web application to be deployed publicly. Hence, hybrid deep learning models form a promising and productive avenue to investigate in NLP based fact verification systems.

5.4 Recommendations

In light of the results of this experiment, identified limitations and the overall trends present in the research literature around fake news detection research, the following recommendations are made to future developers of, researchers into and organizations interested in building on this system:

Integration of Multimodal Signal

The future version of the system could be improved by integrating further signals beyond just the pure text of the article. These include the reliability of the source and their past record of publishing false news, how the article is shared on social media and interaction associated with the article, the history of publication of the author and the inclusion of any accompanying images or video. Evidence has also been gathered indicating that a multimodal system integrating features across text, image, and metadata will significantly and consistently outperform a pure text approach on real-world datasets. Further, a graph neural network has been designed, capable of analyzing the spread patterns of an article on social networks that also show a strong correlation to how fake news articles are disseminated.

Multilingual and Cross-Lingual Extension

This model is currently trained solely on English news media and thus it's of very limited use in terms of predicting real or fake global news that is spreading in every other major language, that exist today. We must future extend our training to substitute bert-based-uncased encoder with one that supports multilingual languages - for example mBERT(multilingual BERT),XLMRoberta support all 100+ major languages in a single model. After further training on such multilingual fake news sets our model can serve to any Nigerian, African region or any country on earth.

Real-Time News URL Analysis

The current method asks the user to Paste the article text into the web app which is inconvenient for actual use. An improvement is suggested that the URL IngestionModule be introduced that can take the URL of an news article, scrape its content(possibly using bs4 or newspaper3k), process the scraped text and make a prediction at a single input by the user. That will allow the user (especially non-tech savvy) or journalist to perform a real-time fact check.

Model Explainability and Attention Visualization

Deep learning classification models are known for their lack of transparency - the user receives a FAKE or REAL label and confidence score, but cannot see which specific words or sentences in the article were responsible for making this prediction. The next stage of development would involve the addition of attention weight visualization; this technique would demonstrate thetokens (in our case, the words or phrases in the input article) that received the highest attention score in the BiLSTM attention pooling stage of operation - enabling the user to

examine the model's thought process. Libraries BERTViz and SHAP (Shapely Additive exPlanations) are available to enable this sort of neural network interpretation.

Continuous Learning and Dataset Expansion

As fake news publishers adopt better detection methodologies, their linguistic styles and tactics shift in response; thus, a model trained once on a particular dataset is unlikely to generalize well to new styles of fake news in the future. It’s best to add a continuous learning pipeline to augment this system. A system could periodically retrain the model based on newly acquired and labelled examples of articles. This pipeline could leverage user prediction accuracy feedback as a “weak supervision” input. Active learning techniques could be used to prioritize selection of only cases on the margin for labelling where the system is not confident.

REFERENCES

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

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1, 4171–4186.

Zhou, X., & Zafarani, R. (2020). *A survey of fake news: Fundamental theories, detection methods, and opportunities*. ACM Computing Surveys, 53(5), Article 109, 1–40.Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171-4186.

Umer, M., Ashraf, I., Mehmood, A., Kumari, S., Ullah, S., & Choi, G. S. (2020). *Sentiment analysis of tweets using a unified convolutional neural network–long short-term memory network model*. Computational Intelligence and Neuroscience.Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1), 1-309.

Horne, B. D., & Adali, S. (2017). This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1).

Kaliyar, R. K., Goswami, A., Narang, P., & Sinha, S. (2021). FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. *Multimedia Tools and Applications*, 80(8), 11765-11788.

Zhao, Y., Zhu, S., Wan, Q., Li, T., Zou, C., Wang, H., & Deng, S. (2022). *Understanding how and by whom COVID-19 misinformation is spread on social media: Coding and network analyses*. Journal of Medical Internet Research, 24(6), e37623.Kim, Y. (2014). *Convolutional neural networks for sentence classification.* Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1746-1751.

Kula, S., Choras, M., & Kozik, R. (2020). *Application of the BERT-based architecture in fake news detection.* International Conference on Computational Science, 239-249.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). *ALBERT: A lite BERT for self-supervised learning of language representations.* International Conference on Learning Representations (ICLR).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach.* arXiv preprint arXiv:1907.11692.

Liu, Y., & Wu, Y. F. (2018). *Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks.* Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B. J., Wong, K. F., & Cha, M. (2016). *Detecting rumors from microblogs with recurrent neural networks.* Proceedings of the 25th International Joint Conference on Artificial Intelligence, 3818-3824.

Manning, C. D., & Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT Press.

Monteiro, R. A., Santos, R. L., Pardo, T. A., de Almeida, T. A., Ruiz, E. E., & Vale, O. A. (2018). *Contributions to the study of fake news in Portuguese: New corpus and automatic detection results.* International Conference on Computational Processing of the Portuguese Language, 324-334.

Perez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2018). *Automatic detection of fake news.* Proceedings of the 27th International Conference on Computational Linguistics, 3391-3401.

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017). *Truth of varying shades: Analyzing language in fake news and political fact-checking.* Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2931-2937.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter.* arXiv preprint arXiv:1910.01108.

Shahi, G. K., & Nandini, D. (2020). *FakeCovid: A multilingual cross-domain fact check news dataset for COVID-19.* Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation.

Sharma, K., Qian, F., Jiang, H., Ruchansky, N., Zhang, M., & Liu, Y. (2019). Combating fake news: A survey on identification and mitigation techniques. *ACM Transactions on Intelligent Systems and Technology*, 10(3), 1-42.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2018). *FakeNewsNet: A data repository with news content, social context and dynamic information for studying fake news on social media.* arXiv preprint arXiv:1809.01286.

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., & Liu, H. (2019). *Beyond news contents: The role of social context for fake news detection.* Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 312-320.

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

Verma, P. K., Agrawal, P., Amorim, I., & Prodan, R. (2021). WELFake: Word embedding over linguistic features for fake news detection. *IEEE Transactions on Computational Social Systems*, 8(4), 881-893.

Wang, W. Y. (2017). *Liar, liar pants on fire: A new benchmark dataset for fake news detection.* Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2, 422-426.

Wardle, C., & Derakhshan, H. (2017). Information disorder: Toward an interdisciplinary framework for research and policy making. *Council of Europe Report,* 27.

Zheng, L., Jin, Z., Luo, J., & Wu, Z. (2022). SVFEND: Cross-modal learning for detecting fake news with entities. *ACM Transactions on Multimedia Computing, Communications, and Applications,* 18(3), 1-20.

Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science, 359*(6380), 1146–1151.