Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

Fake News Prediction Using Developed Machine Learning Models

A Dissertation

Submitted to the Council of the College of Information Technology, University of Babylon in Partial Fulfillment of the Requirements for the Doctor of Philosophy Degree in Information Technology/ Software

By

Tahseen Ali Hassan Hameed

Supervised by

Prof. Dr. Ban Nadeem Dhannoon

2023 A.C. **1446 A.H.**

بسم الله الرحمن الرحيم

۞ فَبَدَأَ بِأَوْعِيَتِهِمْ قَبْلَ وِعَاءِ أَخِيهِ ثُمَّ اسْتَخْرَجَهَا مِنْ وِعَاءِ أَخِيهِ
كَذَٰلِكَ كِدْنَا لِيُوسُفَ مَا كَانَ لِيَأْخُذَ أَخَاهُ فِي دِينِ الْمَلِكِ إِلَّا أَنْ يَشَاءَ
اللَّهُ نَرْفَعُ دَرَجَاتٍ مَنْ نَشَاءُ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ۞

صدق الله العظيم

سورة يوسف الآية 76

Supervisor Certification

I certify that this dissertation entitled **"Fake News Prediction Using Developed Machine Learning Models"** was prepared under my supervision at the Department of Software / College of Information Technology / University of Babylon, by **Tahseen Ali Hassan Hameed** as a partial fulfillment of the requirements of the degree of **Ph.D. in Information Technology / Software**.

Signature:

Name: **Dr. Ban Nadeem Dhannoon**

Title: **Professor**

Date: / / 2023

The Head of the Department Certification

In the view of available recommendations, I forward the dissertation entitled **"Fake News Prediction Using Developed Machine Learning Models"** for debate by the examination committee.

Signature:

Name: **Dr. Ahmed Saleem Abbas**

Title: **Professor**

Date: /   /2023

Declaration

I hereby declare that this Dissertation, submitted to University of Babylon in partial fulfillment of requirement for the degree of Ph.D. in Information Technology \ Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source are appropriately cited in the references.

Signature:

Name: Tahseen Ali Hassan

Date: / /2023

Dedications

To the savior of mankind, Al-Imam Al-Mehdi (peace be upon him), who will fill the earth with justice and equity, after it was filled with injustice and oppression.

To the soul of my father who left me alone since my childhood. May Allah have mercy on you, my father.

To my mother. You have always been a source of kindness and compassion; I grew up on the shoulders of your patience. I got this certificate thanks to your prayers. God bless you for us.

To my wife and children who are the secret of my happiness in life. I thank you for your understanding of my shortcomings during the study period, and I thank your unlimited support, as I live for you.

To my sisters who are the source of tenderness for me. My destiny is to live without a brother, but God blessed me with you.

To every relative or friend who encouraged me and prayed for me from his heart. All respect to you

Acknowledgement

All praise be to ALLAH Almighty who enabled me to complete this task successfully and my utmost respect to his last Prophet Mohammad PBUH and to the hero of Islam born in the heart of Kaaba Amir-Al-Mo'mineen Al-Imam Ali Ibn Abi Talib.

All thanks, appreciation and respect to my supervisor **Prof. Dr. Ban Nadeem Dhannoon**, for her invaluable guidance, supervision and untiring efforts during the course of this work.

I must express my very profound gratitude to my mother and my wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

I would like to declare my deep thanks and appreciation to my sisters for their patience, encouragement, and help during the work.

Special thank goes to the staff of the University of Babylon, especially the staff of the Presidency and staff of the College of Information Technology for the great help they have offered me.

Last but not least, but most importantly, I would like to thank all the kind, helpful and lovely people who helped me directly or indirectly to complete this work and apologize to them for not being able to mention them by name here, but they are in my heart.

Tahseen Ali Hassan

Abstract

Recently, the world has been witnessing the so-called age of social media, as there is a complete dependence on these sites to follow up on events and activities. On the other hand, most news agencies follow the wills of their founders or owners, which makes people unbelieving the credibility of any news or article. The problem is that fake news always focuses on critical topics and is released appropriately, so it spreads quickly and takes a wide resonance. As such, the research community has contributed to conducting many studies to reduce this phenomenon. Although several studies were performed to detect English fake news, the identification of Arabic misinformation remains underdeveloped. This dissertation aims to build developed machine learning models for fake news detection in Arabic and English. Fake news predictions are improved through three developed models: Developed Machine Learning Model (DMLM), the Attention-based Bidirectional Hybrid Deep Model (ABHDM), and the Ensemble Deep Learning Model (EDLM).

The Fuzzy Model (FM) is employed to select the relevant user-based features. Initially, these features are evaluated by two proposed methods: Mean Decrease in Gini-index (MDGI) and Drop-Feature Importance (DFI). User-based features have been evaluated a third time using the correlation method. Finally, the FM is fed with the outputs of these methods to identify the relevant features. As for the DMLM, the random forest classifier has been improved by focusing on the relevant user-based features in building trees instead of random selection.

Two common text-processing techniques are Bidirectional Long Short-Term Memory (Bi-LSTM) and Bidirectional Gated Recurrent Unit (Bi-GRU). Although the popularity of these networks in handling arbitrary

sequences efficiently, they treat features extracted from the text with equal importance. Consequently, the ABHDM is proposed in this dissertation to address this problem. In this model, a convolutional neural network (CNN) is applied to reduce the dimensionality of a dataset. Then, two independent Bi-LSTM and Bi-GRU layers are used. The idea behind the individual implementation of these layers is to process short and arbitrary texts at the same time. The attention model is applied to the outputs of these layers to identify more informative words. Finally, multi-layer perceptron (MLP) is applied to classify the news article as fake or real. The EDLM, on the other hand, has used each of Bi-LSTM, Bi-GRU, and CNN to determine the credibility or not the credibility of any news article. This model is built similarly to the ensemble approach commonly used in machine learning. Therefore, the final decision in the classification of the news article is taken based on the opinions of all these networks.

Experiments have been conducted on five datasets: Arabic Twitter Dataset, Sentimental Lair Dataset, AraNews Dataset, Arabic Fake News Dataset (AFND), and Fake-or-Real Dataset. Accuracy has been used to measure the quality of our proposed models, whereby the accuracy of DMLM reached 0.886 and 0.762 for the Twitter and Sentimental LAIR datasets, respectively. In contrast, the accuracy of ABHDM reached 0.8614, 0.9447, and 0.9944 for the AFND, AraNews dataset, and Fake-or-Real dataset, respectively. The prediction results of the EDLM are slightly lower than the results of ABHDM in terms of performance metrics. Moreover, the results of our proposed models are better compared to previous studies that used these datasets.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Abbreviation	Meaning
ABHDM	Attention-based Bidirectional Hybrid Deep Model
Adam	Adaptive Moment Estimation
AFND	Arabic Fake News Dataset
BERT	Bidirectional Encoder Representations from Transformers
Bi-GRU	Bidirectional Gated Recurrent Unit
Bi-LSTM	Bidirectional Long Short-Term Memory
CNN	Convolution Neural Network
COG	Center of Gravity
DFI	Drop-Feature Importance
DL	Deep Learning
DMLM	Developed Machine Learning Model
EDLM	Ensemble Deep Learning Model
FM	Fuzzy Model
FN	Fake News
FND	Fake News Detection
GloVe	Global Vector
LR	Logistic Regression
MDGI	Mean Decrease in Gini-index
ML	Machine Learning
MLP	Multi-Layer Perceptron
MF	Membership Function
NLP	Natural Language Processing
ReLU	Rectified Linear Unit
RF	Random Forest
RMSProp	Root Mean Squared Propagation

RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
Tanh	Hyperbolic Tangent
TF-IDF	Term Frequency-Inverse Document Frequency

List of Dissertation-Related Publications

(First Paper)

Name of Journal: Karbala International Journal of Modern Science.

Paper Title: Improving Prediction of Arabic Fake News Using Fuzzy Logic and Modified Random Forest Model.

Scopus: Q1 (Free Fees) Cite Score = 4.8

Authors:

- Tahseen A. Wotaifi.
- Ban N. Dhannoon

(Second Paper)

Name of Journal: journal of universal computer science.

Paper Title: Developed Models Based on Transfer Learning for Improving Fake News Predictions.

Clarivate - Impact Factor 1.056, Scopus Q2 (Free Fees)

Authors:

- Tahseen A. Wotaifi.
- Ban N. Dhannoon

(Third Paper)

Name of Journal: Baghdad Science Journal.

Paper Title: An Effective Hybrid Deep Neural Network for Arabic Fake News Detection.

Scopus Q3, Cite Score = 0.9

Authors:

- Tahseen A. Wotaifi.
- Ban N. Dhannoon

(Fourth Paper)

Name of Journal: Iraqi journal of science.

Paper Title: Attention Mechanism Based on a Pre-trained Model for Improving Arabic Fake News Predictions.

Scopus: Q3, Cite Score = 0.8

Authors:

- Tahseen A. Wotaifi.
- Ban N. Dhannoon

(Fifth Paper)

Paper Title: ABHDM: Attention-Based Hybrid Deep Model for Improving Fake News Predictions. **[Under review in The Journal of Supercomputing: Springer (Clarivate - Impact Factor 2.850), Scopus Q2 (Free Fees)]**

Note: This paper was accepted in the International Journal of Electrical and Computer Engineering (Scopus Q2), but we requested the editor of the journal to withdraw from publication and sent it to the above journal**.**

Authors:

- Tahseen A. Wotaifi.
- Ban N. Dhannoon

Chapter One

General Introduction

General Introduction

1.1 Introduction

Fake News (FN) is information or a claim verified to be incorrect [1]. Throughout the last few years, FN has been a common research topic in data mining fields and Natural Language Processing (NLP) [2]. Historically, the roots of FN go back to the 17th century in the form of propaganda that later turned into disinformation. Today, this phenomenon refers to the publication via the Internet of false information, intentionally or unintentionally [3]. Social media is another source of exchanging misinformation between people. A commonly used social media platform is Twitter, which millions of users adopt. The reason behind the tendency of online users to use this platform is that it provides a channel that allows users to easily create, publish and share information [1][4]. Usually, FN is produced by so-called fakesters, who create an article with false content, often injected with trusted news content [5].

The probability that an article, story, tweet, or publication is intentionally misleading is called Fake News Detection (FND) [6]. As a result of the lack of censorship on the Internet in news publications and the huge amount of information circulating among people, FND has become a difficult task [7]. The task of FND is further complicated by the presence of many fakesters who aim to deceive the user for several purposes, including religious, political, etc. Another source of fake news, as confirmed by the Poynter institute, is that satirical news often turns into fake news that floods the web [8]. Other terms close in concept to FND are rumour detection, disinformation classification, stance classification, and misleading information detection [9]. In the last few years, researchers and the scientific community have paid great attention to FND. The research community widely uses artificial intelligence technologies, including machine learning (ML) and deep learning (DL) to identify the credibility of news articles [10].

Manually determining whether a news article is fake or real is slow, tedious, and impractical. A promising solution proposed by many studies is the use of DL and ML algorithms for FND [11]. The ML models usually apply traditional classification methods such as Random Forest (RF), Logistic Regression (LR), Naïve Bayes, and others on linguistic features. These features are extracted by several methods such as Bag of Word (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), etc., which have two major drawbacks; (1) it takes a long time to represent a word into a feature; (2) words that are similar or close in terms of letters are assigned somewhat similar vectors, which is the illusion of an ML model [12]. To handle the above drawbacks, word embedding techniques were proposed and used by many previous studies. The idea behind these techniques is to represent words based on semantic meaning, making them the input to DL [13].

The DL has achieved high performance in many fields, including sentiment analysis, computer vision, and image processing [14]. Among the wide deep neural network types, GRU, LSTM, and CNN are common in sequence processing [15]. Both GRU and LSTM are suitable for processing long and arbitrary sequences. Then, Bi-GRU and Bi-LSTM were used to consider succeeding and preceding contexts by combining backwards and forward hidden layers [16]. Although the popularity of these networks, there have two major drawbacks: (1) the high dimensional space makes the network more difficult to improve; (2) GRU and LSTM cannot focus on important information in the text as they give all words equal importance [17]. Thus, this dissertation has focused on identifying the most informative by using CNN and attention model.

On the other hand, the dissertation has contributed to building an FM for identifying relevant user- and content-based features. Based on the relevant selected features, the random forest classifier has been developed by focusing on these features in building trees instead of random selection.

1.2 Problem Statement

The FN has become an important discussion topic in society because of the permanent dangers this news poses to individuals, communities, and countries. FND is a natural language processing task that involves identifying and classifying news articles or other types of text as real or fake..With the absence of censorship on the Internet and the huge amount of data exchanged daily, FND is a considerable challenge. Another difficulty is that publishers (fakesters) usually choose sensitive topics and publish them at critical times, so their publications have a wide resonance and spread very quickly. On the other hand, most news agencies follow the opinions and whims of their owners, so the news issued by these agencies was in doubt for the recipient.

Manually identifying news articles as fake or real is difficult, tedious, and impractical. In other words, identifying FN and stopping its impact on societies is difficult for humans because it requires a complicated activity to collect evidence and sift the facts before making a decision. For all the preceding, this dissertation is concerned with proposing the appropriate steps that can be taken to provide sound solutions to the FN problem.

1.3 Related Works

Many studies have tried to solve the problem of FN and limit its spread by building prediction models. Some studies have discussed the challenges of FND from the perspective of NLP and data mining. Some of these approaches consider content-based features and user-based features with ML to classify news articles. In contrast, Recurrent Neural Network (RNN) is the first deep learning technique considered to solve this problem. This network improved its predictions by deriving other versions such as LSTM and GRU.

Recently, researchers have contributed to a new development of NLP by building so-called pre-trained models (e.g., Global Vector (GloVe), FastText, etc.) for word representation. For all of the above, we present some of the studies that strived to reduce the problem of FN. All the previous studies listed below were conducted on the datasets used in our dissertation, and the results found that our proposed models outperformed these studies.

In [8], experiments were conducted on the Fake-or-Real dataset to detect English fake news. Initially, words were converted into numerical features through traditional text representation methods. Then, more than one ML technique was used to classify news articles. The Support Vector Machine achieved the best performance among the prediction models used in this work.

In [1], the authors have utilized ML to identify misinformation or unreliable tweets in Arabic based on a supervised classification model. This study is efficient in that it included both content-based and user-based features, but the accuracy of the models was not sufficient. The random forest classifier achieved the best results among the ML techniques used in this study.

As for the study by [7], the authors built the models based on the Sentimental LIAR dataset for English fake news detection. A CNN based on BERT base pre-trained model was proposed to detect English fake news. Although the study presented a hybrid and valuable method, the results were not up to the level of ambition.

The authors in [2] have used Twitter to collect the tweets dataset. The study proposed a new feature selection method using optimization techniques to select the best user-based features. On the other hand, many text representation methods, such as TF-IDF and BoW, were used to convert words into numerical features. Finally, ML techniques were applied to classify any tweet as credible or not credible. Logistic Regression has achieved the best results among the prediction methods used in this study.

The study in [4] has used DL for Arabic fake news detection based on four datasets, including the AraNews dataset. In this study, pre-trained models were used for word representation. After that, a deep neural network was built based on CNN to classify news articles. The experiments conducted on the AraNews dataset showed that the AraBERT pre-trained model with CNN achieved the best results.

As for the study in [3], the authors utilized the AraNews dataset to construct the prediction models. In word representation, TF-IDF was used to convert each word into a numerical feature. Then, Naive Bayes, Random Forest Classifier, and Logistic Regression are applied for Arabic fake news detection. Among these models, the random forest classifier has achieved the best results.

In [6], the authors have used AFND to build prediction models. The study focused on both DL and ML for Arabic fake news detection. In DL, an embedding layer was used for word representation, whereas ML models relied on traditional methods such as TF-IDF in feature extraction. Through experiments, it was found DL models achieved higher accuracy compared to ML models. Moreover, multi-classifier and binary-classifier were performed, and in both approaches, the capsule network model achieved the best accuracy (0.709 Accuracy for multi- classifier and 0.79 Accuracy for the binary classifier).

In [5], the authors conducted their experiments on AraNews and other datasets. Initially, the study focused on word embedding methods (pre-trained models) for word representation. On the outputs of these methods, an artificial neural network was added to classify news articles. JointBERT pre-trained model with a batch size equal to 32 has achieved the best results.

A summary of the previous studies with additional details is listed in Table (1.1):

Table (1.1) Summary of Related Work

Year	Previous Study	Title	Dataset	Techniques	Best Results
2018	[8]	Detecting opinion spams and fake news using text classification	Fake-or-Real Dataset	Support Vector Machine	Acc. = 0.92
2019	[1]	Classifying Arabic Tweets Based on Credibility Using Content and User Features	Tweets Dataset	Random Forest	Acc. = 0.76 Pr. = 0.79 Re. = 0.82
2020	[7]	Sentimental LIAR: Extended Corpus and Deep Learning Models for Fake Claim Classification	Sentimental LIAR	CNN	Acc. = 0.70 F-score = 0.65
2021	[2]	Intelligent Detection of False Information in Arabic Tweets Utilizing Hybrid Harris Hawks Based Feature Selection and Machine Learning Models	Tweets Dataset	Logistic Regression	Acc. =0.815 Pr. = 0.8552 Re. = 0.8235
2021	[4]	Arabic Fake News Detection: Comparative	AraNews Dataset	CNN	Acc. = 0.8002 Pr. = 0.8004

		Study of Neural Networks and Transformer-Based Approaches			Re. = 0.9991
2021	[6]	Detecting Arabic Fake News Using Machine Learning	AFND	capsule network model	Acc. = 0.79
2022	[3]	Multi-scale Machine Learning Prediction of the Spread of Arabic Online Fake News	AraNews Dataset	Random Forest	Acc. = 0.866
2022	[5]	JointBert for Detecting Arabic Fake News	AraNews Dataset	Hybrid Model	Acc. = 0.80

1.4 Challenges of Dissertation

Many research efforts were accomplished to identify English fake news on the Internet or social media. However, developing a deep model for detecting fake Arabic news proved to be a challenging task for several reasons: (1) lack of resources; (2) most NLP tools do not support the Arabic language; (3) instead of the formal Arabic language, most people tend to utilize the informal language.

On the other hand, collecting tweets from Twitter and labeling them requires more effort. Due to the noisy nature of Twitter content, data cleaning and processing takes a lot of time. At the same time, labeling tweets by the expert requires extra time and focus.

In addition, the biggest challenge is that no appropriate dataset in the Arabic language is available to the research community. Even with the availability of the dataset, dealing with FN requires constant updating of that dataset. The reason for this is that increasing events in most countries may lead to updating an article or news from fake to non-fake, or vice versa.

1.5 Dissertation Aims

This dissertation aims to improve the prediction of any news article, story, publication, or tweet as fake or real. To achieve this aim, the following objectives will be considered.

1- Constructing a developed feature selection models to identify relevant user-based features and content-based features.

2- Building developed machine learning model for improving the prediction of Arabic fake news published on Twitter.

3- Proposing a new architectures of deep neural networks to improve prediction accuracy through its ability to address arbitrary and short news articles.

1.6 Dissertation Contributions

This dissertation has achieved the following contributions in comparison to previous work:

1- Adapting Bidirectional Encoder Representations from Transformers (BERT) pre-trained model with our proposed machine learning model by extracting the embedding sentences vector. Most of the previous studies add an output layer to this model to perform classification tasks, while this dissertation focused on the outputs of this model and employing them with DMLM.

2- Two feature selection methods are proposed to identify the relevant user-based feature (MDGI and DFI). Based on the outputs of these methods, a reliable feature selection method using fuzzy logic was proposed.

3- The behaviour of the random forest classifier was modified and improved by focusing on the relevant features in constructing trees rather than random selection.

4- A deep neural network is proposed to predict arbitrary and short news articles. This network is hybridized from three networks (CNN, Bi-LSTM, and Bi-GRU) with the participation of the attention mechanism.

5- Building a new architecture for deep neural networks based on the ensemble approach. This architecture produces its predictions by taking the opinion of three common deep learning algorithms.

1.7 Dissertation Organization

After this introductory chapter, the rest of the dissertation is structured as follows:

Chapter Two, "Theoretical Background": Many concepts will be introduced in this chapter, including datasets used in this dissertation, ML, feature selection, fuzzy logic, NLP, transfer learning, DL, and evaluation measures.

Chapter Three, "Methodology and Proposed System": This chapter first views the proposed methodology. Then, two parts will be discussed. The first part presents developed ML. Furthermore, the second part explains the proposed hybrid deep neural networks.

Chapter Four, "Results and Discussion": This chapter presents the results of the proposed models on all the datasets used in this dissertation. Then, the results will be compared with the previous studies conducted on these same datasets.

Chapter Five, "Conclusions and Future Works": Many conclusions drawn from our proposed models and suggestions for future work will be introduced.

Chapter Two

Theoretical Background

Theoretical Background

2.1 Overview

This Chapter introduces an overview of modern text analysis and processing concepts. Furthermore, it discusses the datasets used in this dissertation and then explains the main steps in classifying texts. Feature selection, word representation, and fuzzy logic are also highlighted. Next, common ML techniques for FND were demonstrated. Finally, this Chapter shows the theoretical background of the key concepts of DL and the metric performance in classification algorithms.

2.2 Datasets

Data collection is the first stage or starting point in any ML or DL project. Most previous studies in the text mining field rely on social media, websites, or news agencies for data collection. When data is collected, the research community faces a difficult challenge: data labeling. Two main reasons behind the difficulty of data labeling: (1) The dataset is labeled by experts (this procedure is accurate but tedious and time-consuming); (2) The data is labeled by semi-supervisor learning (this procedure is fast, but it is prone to some errors) [18]. This dissertation focuses on addressing FN in both Arabic and English and uses five datasets: the Twitter dataset, AraNews dataset, AFND, sentimental LAIR dataset, and Fake-or-Real dataset.

The Twitter dataset collected by the authors in [1] is used in this dissertation. Twitter API enables users to access and download tweets and a set of user-based features (the writer of the tweet) and other features of the tweet itself [2]. As such, this dataset was expanded from 1,862 tweets to 3,000 tweets with the same features used by the researcher, as illustrated in Appendix A. After that, the tweets are labeled manually by the experts.

The second dataset used in this dissertation is called AraNews dataset and is available at kaggle.com. The AraNews dataset is a large and public Arabic fake news dataset collected from many newspapers on many topics such as politics, economics, sport, etc. This dataset was collected from 15 Arabic countries, the United States of America, and the United Kingdom [3]. No additional features were used in this dataset, so NLP was performed on the textual content of each news article [4].

AFND is a newly launched Arabic dataset. This dataset is suitable for DL tasks because it contains a large number of news articles. The exact number of news articles in AFND is 606912, collected from 134 sources distributed in 19 Arab countries. The news articles are labeled as credible, undecided, or not credible [6].

LIAR is a short statement dataset containing five target classes: false, pants-fire, barely-true, mostly-true, and true. Sentimental LIAR is a modified version of the LIAR dataset converted into binary-class labeling [7]. In addition to texts, this dataset contains features related specifically to the speaker or writer, as shown in Appendix B.

The Fake-or-Real dataset was collected from real-world sources. The real news in this dataset was collected from an authoritative news site called Reuters.com. As for the fake articles, it was collected from a fake dataset on kaggle.com. For each article, some information is available such as the article title, article type, article date, and class label [8].

2.3 Feature Selection

The main objective of feature selection is to select the best features (a relevant subset of features) among all the features. This reduces time complexity (reducing dimensionality) and improves classifiers by excluding irrelevant features. Generally, three common feature selection approaches are filter, embedded, and wrapper [19].

These methods identify the best features based on their importance to the target class [20]. All the filter methods assign a weight to each feature based on its impact on the prediction process, and they are always applied before the prediction model, as shown in Figure (2.1).

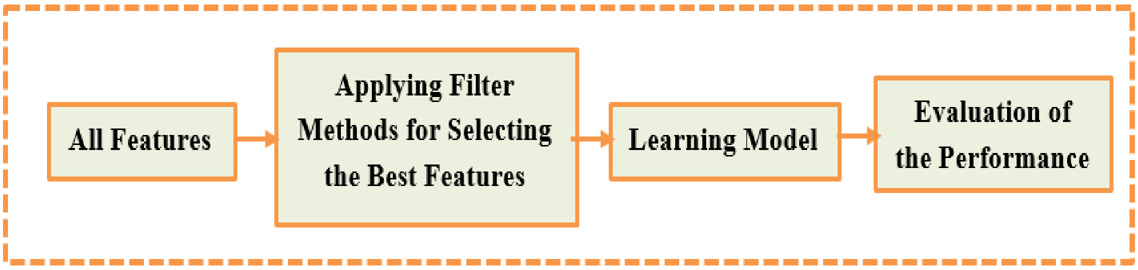


Fig. (2.1): Filter Method Framework [21]

The correlation method is one of the important filtering methods. This method evaluates the features based on their correlation with the target class. The feature weight is between (-1) and (1), so the feature is uncorrelated with the target class (weak) if its weight is equal to zero, whereas it is strong if its weight is close to ±1. The correlation value is calculated by Equation (2.1) [22].

$$\mathrm{cor}(x, y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X-\bar{X})^2}\sqrt{\sum(Y-\bar{Y})^2}} \quad (2.1)$$

Where: X and Y represent the value of feature and target class respectively and $\bar{X}$ and $\bar{Y}$ are the mean

The embedded methods are a special type of feature selection as they are executed during the learning model. Decision trees only implement these methods, where the features are pruned during the training process [23]. In contrast, the wrapper methods differ from the filter and embedded methods, as they identify the best subset of features by their performance in the prediction model. These methods use the strategy search to identify the relevant features [24].

2.4 Text Pre-processing

Text pre-processing is the process of converting texts or sentences into an appropriate form for the classifier model. Text pre-processing aims to reduce data's feature space or dimensionality. Many pre-processing steps are conducted for text classification, such as removing punctuations, symbols, numbers, etc. After the text is cleaned, the tokenization approach is performed [25]. Word tokenization divides the text document into words (tokens) based on spaces between these words. Usually, two important pre-processing tasks are performed with ML: removing stop words and stemming. The main benefit of these tasks is to reduce the dimensionality of the data by removing unimportant words and returning the words to their roots [26].

2.5 Text Feature Extraction

The task of converting a particular text into feature vectors is called feature extraction [27]. The outputs of this process are numerical vectors, so they are often called vectorization. The extracted features from the text are fed into the prediction model in order to perform text classification [28]. Previously, many studies used traditional methods such as TF-IDF for word representation. Nowadays, the research community focuses on embedding methods to represent words because these methods allocate a vector for each word based on the semantic meaning of that word [29].

2.5.1 Term Frequency-Inverse Document Frequency

TF-IDF is one of the feature extraction methods that defines the important words in a set of articles or documents. This method assigns weight or score to each word based on its appearance in articles or documents

[30]. As shown in Equation (2.2), TF-IDF focuses on Term Frequency (TF) and Inverse Document Frequency (IDF) to identify the score of each word [31].

$$Score: TF - IDF\ (t,d) = TF\ (t,d) * IDF\ (t) \qquad (2.2)$$

Where (t) represents the word (term), and d means a document. TF is the number of times a term t occurs in document d. In other words, TF measures the frequency of a word or term (t) in document (d) and is calculated according to Equation (2.3) [31].

$$TF\ (t,d) = \frac{count\ of\ t\ in\ d}{number\ of\ words\ in\ d} \qquad (2.3)$$

IDF measures the informativeness of the term (t) in a set of documents and is calculated according to Equation (2.4) [31].

$$IDF\ (t) = \log_2 \left(\frac{N}{(\ df + 1)}\right) \qquad (2.4)$$

Where (*df*) is the number of occurrences of the term t in a set of documents N (number of documents in which the term t is present). IDF is usually very low for the most occurring words (e.g., stop words) because these words are apparent in all documents. Division by zero was avoided for words that did not appear in the documents by adding 1 to the denominator.

2.5.2 Word Embedding

A word embedding is a learned representation for texts where words that have the same meaning have a similar vector (similar representation) [32]. Word embedding techniques are an improvement over the traditional TF-IDF model, where large sparse vectors were utilized to represent the words. Instead, in embedding techniques, each word is represented by dense vectors by suggesting a set of features or criteria. The embedding values in the word vector represent the proximity of that word to the proposed criteria. The process of manually constructing the embedding matrix for vocabulary is tedious, difficult, and time-consuming. To overcome this challenge, two popular and efficient methods can be used: (1) a pre-trained model (see the next section); (2) a Keras embedding layer [33].

The Embedding layer is the first hidden layer of a deep neural network introduced by Keras for processing the texts. The input data of this layer is integer encoded, where a unique integer is assigned to each word (Keras provide Tokenizer API to implement this task). The output of this layer is an embedding matrix that is initialized randomly, and then the embedding values of this matrix are updated along with the neural network model itself [34]. Three arguments control the size of the embedding matrix: (1) input_dim, which represents the size of the vocabulary; (2) output_dim, which means the length of the vector to each word; (3) input_length, which represents the size (length) of input sequences [35].

Figure (2.2) shows how to obtain the embedding vectors for a two-word sentence (nice food) using Keras embedding layer. In this *example* the arguments are defined as follows: input_dim = 5000, output_dim = 4, and input_length = 2

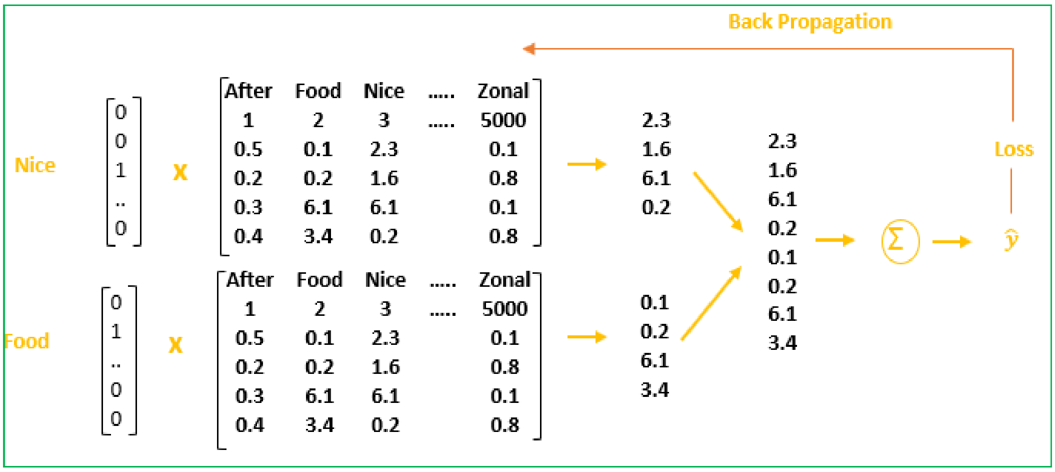


Figure (2.2): Steps of Embedding Layer

2.6 Transfer Learning

As the name implies, transfer learning is transferring what you have learned from one task to another. Transfer learning or so-called pre-trained models were proposed for word representation. Pre-trained models are trained on huge datasets to learn a certain task, saved, and then utilized for solving other similar tasks [36]. These models are a recent trend in NLP and are characterized by their efficiency for two reasons: (1) they solved the problem of a small dataset; (2) no need to train the model from scratch [37]. GloVe, Bidirectional Encoder Representations from Transformers (BERT), and FastText are three common pre-trained models on Arabic and English datasets.

GloVe pre-trained model is an unsupervised learning algorithm developed as an open-source project at Stanford. The effectiveness of this model lies in representing words and assigning a vector to each word through its semantic meaning. The main idea behind GloVe is to adopt a co-occurrence matrix for identifying the relationship between the words; thus, semantically similar words are usually assigned similar vectors [38]. A co-occurrence matrix focuses on pair of words that appear together and its values are a count of these pairs when occurring together. The output of the GloVe is either a matrix of vectors of size 300 or a matrix of vectors of size 100 for each vector. The general form of the GloVe model is given by Equation (2.5) [39].

$$F\left(w_i, w_j, \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{2.5}$$

Where i and j are word vectors, and k represents the context vector. $\boldsymbol{P_{ik}}$ is the probability of appearing word i with context k and $\boldsymbol{P_{jk}}$ is the probability of appearing word j with context k.

FastText is another pre-trained model developed by Facebook for text classification tasks and word embedding learning. The FastText pre-trained model is a simple neural network that uses only one layer for word representation. Instead of assigning vectors for words directly, this model represents the word as an n-gram of characters [40]. *For example*, in the word "technology" with n=3, the representation of this word is <te, tec, ech, chn, hno, nol, olo, log,ogy, gy>, where the angular brackets represent the end and beginning of the word. Once the word is represented using n-grams, a model is trained to learn the embeddings.

FastText is efficient with rare words or tokens; thus, even if any word is not seen during training, it can be broken down into n-grams to obtain its embeddings. In other words, FastText can represent words, not in the dictionary. Through FastText, each word was represented with an embedding dimension equal to 300 [41].

BERT is an open-source deep-learning framework for NLP that was pre-trained using Wikipedia data. Most pre-trained models identify the context by reading the text (sentence) in one direction, but the BERT pre-trained model is different in that it is designed to focus on both directions at once. Unlike other embedding methods such as GloVe and FastText that map each word to a vector (encoder-decoder model), BERT's model considers all the other words in the sentence (text) rather than processing them one by one. As a result of this property, this model was used for many tasks, such as text classification, sentence similarity, and missing word prediction [42].

Two important versions of BERT: BERT_BASE with 12 Layers and BERT_LARGE with 24 layers. For classification, BERT_BASE takes the CLS token (classification token) as the first input, followed by the words of the sentence. Self-attention is applied in each layer, and the results are passed to the next layer. The model's output in each hidden state is a vector of size 768, where the vector corresponding to the CLS token represents the whole sentence that can be used for classification tasks [43]. BERT Architecture is explained in Figure (2.3).

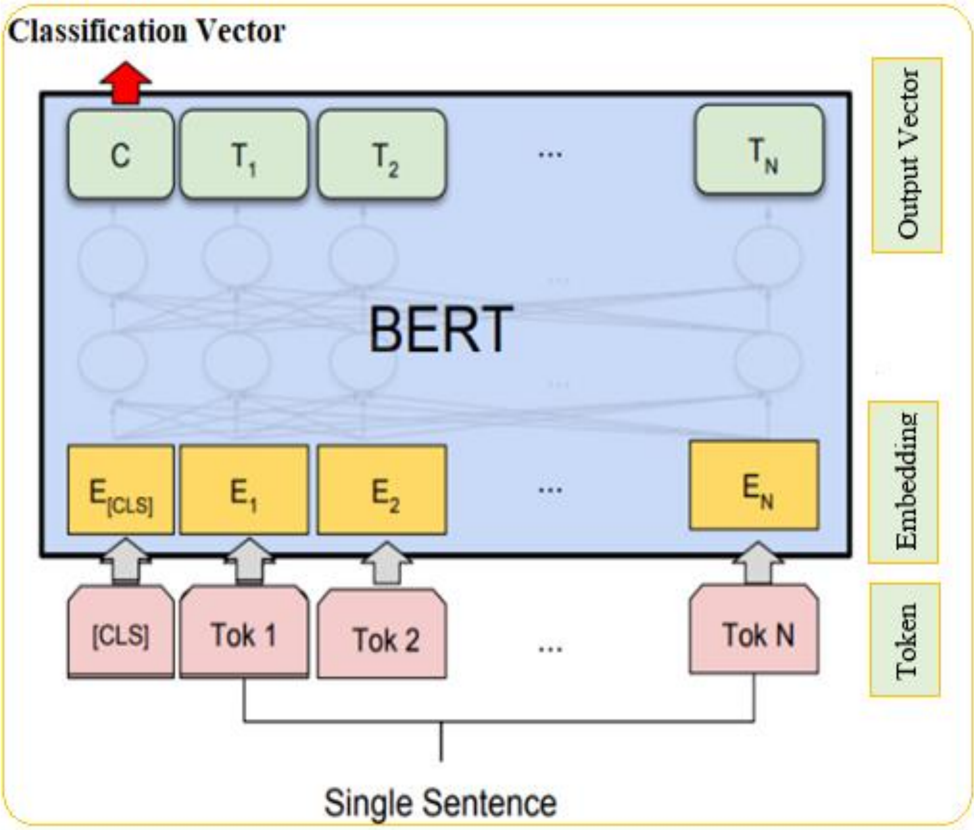


Figure (2.3): BERT Architecture [43]

2.7Attention Mechanism

The Attention Model represents the latest development that scientists have developed in the field of NLP so it is called "attention is all you need". The idea of the attention model is to assign different weights to words in a sentence (news article). The attention model addresses the entire sequence (text, sentence, or article) and summarizes the more informative words into a fixed-length context vector. Therefore, this model gives more attention to certain words in the text and ignores other words [44]. As shown in Figure (2.4), the architecture of the attention mechanism consists of three layers:

the encoder layer, the attention layer, and the decoder layer. Usually, LSTM or GRU are applied in the encoder and decoder layers. In contrast, the main role of the attention layer is to generate a context vector. This is achieved through three different processes: alignment process, softmax calculation, and context vector calculation [45].

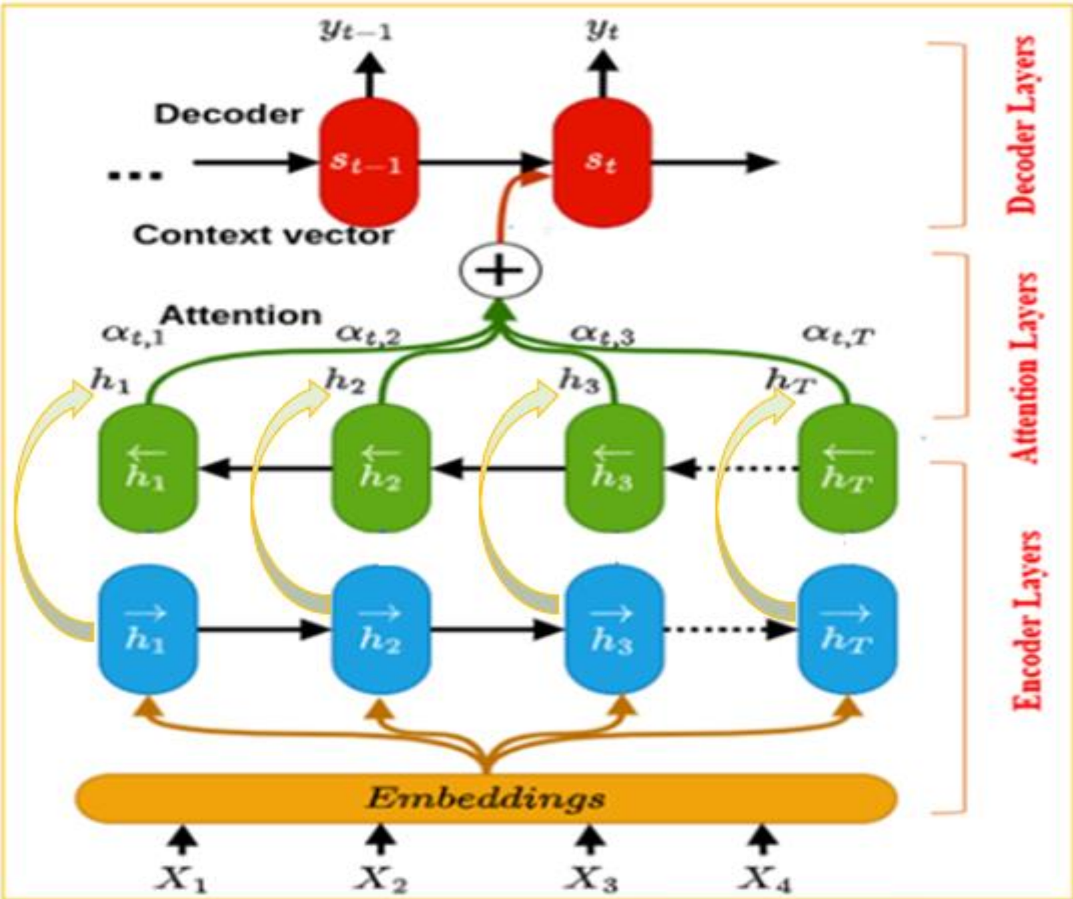


Figure (2.4): Attention Model Architecture [45]

In the alignment process, the feedforward neural network is implemented using each input vector's activation function (f). As explained in Equation (2.6), the input of this function is the encoded hidden states (h_t) and the previous decoder output (s_{t-1}) [46].

$$e_{t,i} = f(s_{t-1}, h_t) \qquad (2.6)$$

As shown in Equations (2.7) and (2.8), the softmax operation is applied to the outputs of the previous step (alignment process). Then, the context vector is fed into the decoder at each time step. Finally, a unique context vector is computed by a weighted sum of all encoder hidden states (T) to focus on the more informative words [46].

$$\alpha_{t,i} = softmax\left(e_{t,i}\right) \tag{2.7}$$

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} \,.\; h_t \tag{2.8}$$

2.8 Fuzzy Logic

In classical (traditional) logic, an element (x) is either a member or non-member of (A). Therefore, the membership function of this element can be expressed through the following formula [47].

$$\mu_A = \begin{cases} 1 & if\ x \in A \\ 0 & if\ x \notin A \end{cases} \tag{2.9}$$

On the other hand, in fuzzy logic, a value or weight between (0 - 1) is assigned to each element through Membership Functions (MF). These values are this element's degree of belonging or importance in the fuzzy set [48].

In fuzzification, the Triangular MF is one of the most important MFs that convert clear values into fuzzy values. This function is described in Figure (2.5) and Equation (2.10) [49].

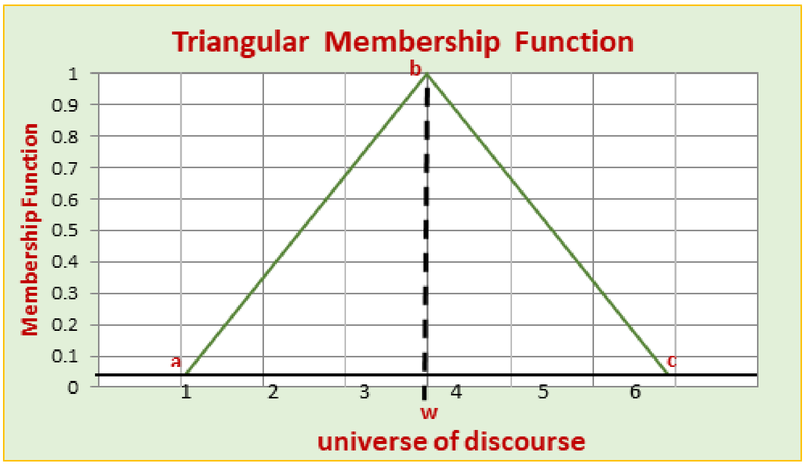


Figure (2.5) Triangular Membership Function [49]

$$\text{Triangular MF } (x{:}\ a, b) = \frac{w-a}{b-a} \qquad (2.10)$$

Where: w is the real value within the universe of discourse,whereas a and b are the coordinates of the triangular heads.

In defuzzification, the Center of Gravity (COG) is the function applied to each fuzzy value (resulting from the MF) to obtain a final and clear value. The COG is represented by Equation (2.11) [49].

$$\mu o = \frac{\sum_{i=1}^{n}(\mu(Xi) * Xi)}{\sum_{i=1}^{n}\mu(Xi)} \qquad (2.11)$$

Where: $\mu(\boldsymbol{Xi})$ is MF of element X.

2.9 Machine Learning for Text Classification

ML is an important area of artificial intelligence concerned with building and understanding models that learn from data. The ML concept involves many techniques that build the model on a sample of data called a training set. Then the quality of the model or algorithm is measured by predicting the second part of the data, called the testing set. From these datasets and by applying machine learning techniques, important information and hidden patterns can be extracted within the concept of data mining [50]. When data is text or documents, this concept is called text mining. Text classification is the primary task in the concept of text mining and is one of the popular tasks in NLP that separates texts into a pre-defined class for each document, article, or tweet. These classes can be based on sentiment, topic, or genre [51]. Figure (2.6) illustrates the framework of the text classification.

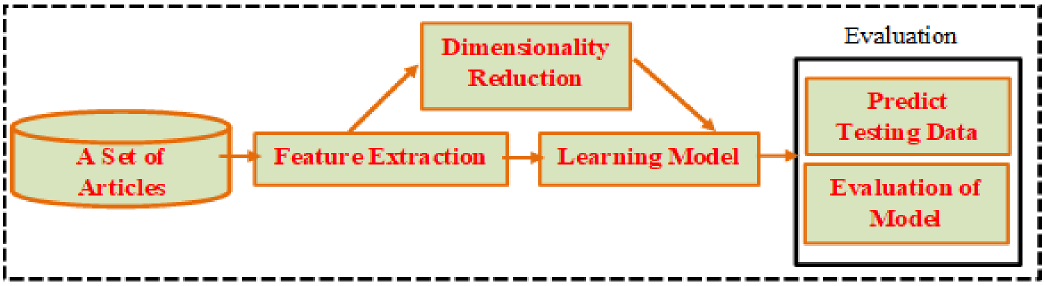


Figure (2.6): Text Classification Framework [51]

2.9.1 Random Forest

Random Forest (RF) is an ensemble technique that makes the final decision based on a set of decision tree models. Each decision tree in the random forest is constructed by randomly taking a subset of the features from the original dataset. The number of features considered in each subset is according to Equation (2.12) [52].

$$F = log_2 f + 1 \qquad (2.12)$$

Where: F is the number of selected features, and f is the number of all features in the dataset.

The final prediction calculation depends on the task type, whether classification or regression. For the classification task, prediction is performed by taking a majority voting of all predictions made by each decision tree model [53]. The idea of RF is presented in Figure (2.7).

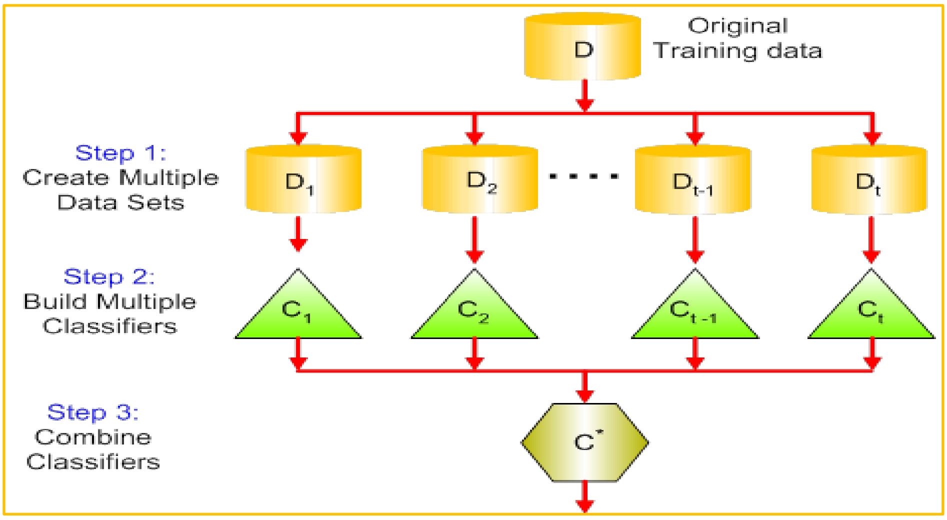


Figure (2.7) RF Framework [53]

2.9.2 Logistic Regression

Logistic Regression (LR) focuses on the relationship between the categorical variable (the dependent variable or the so-called target class) and a set of independent variables (features) [54]. In contrast to linear regression, which is used with continuous values, logistic regression can be used with discrete values as it treats data with a different distribution and shape. Figure (2.8) shows a simple example of a data distribution that can be processed by logistic regression using the sigmoid function, whereas the dependent variable (target class) is described by Equation (2.13) [55].

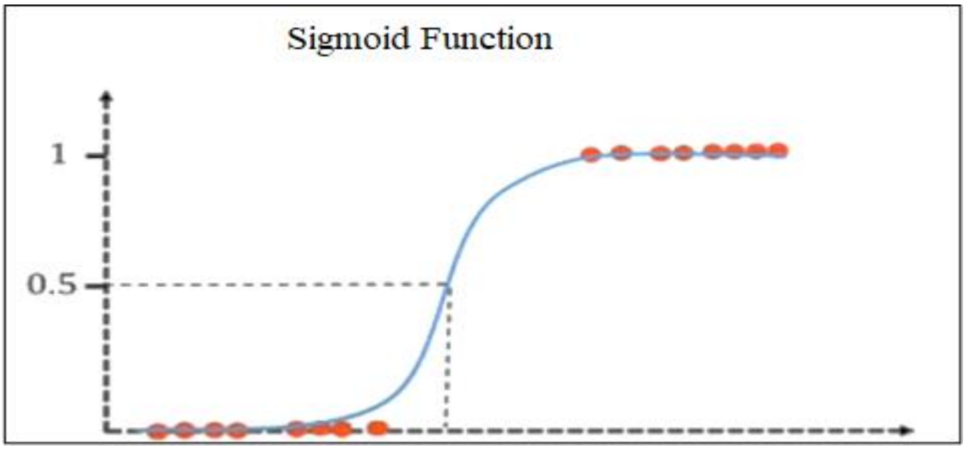


Figure (2.8): LR using the Sigmoid Function [55]

$$Y = \frac{1}{1 + e^{-x}} \qquad (2.13)$$

2.9.3 Multi-Layer Perceptron

MLP is considered one of the most common fully connected class of feedforward artificial neural networks that efficiently learn complex systems. In MLP, at least three layers of neurons: (1) input layer; (2) hidden layer and (3) output layer. Except for the neurons of an input layer, each neuron in the hidden layer and output layer uses a nonlinear activation function to distinguish data points that are not linearly separable. For training, this network uses a supervised learning approach called backpropagation (see section 2.13) [56].

Figure (2.9) shows the three layers of MLP. The input layer (x_0,...,x_n) is the first layer that inputs data to feed the network's neurons. The hidden layer (a_1,…,a_m) is the second layer responsible for data processing, while the output layer (a_1,…,a_k) is the third layer concerned with the output of the results [57].

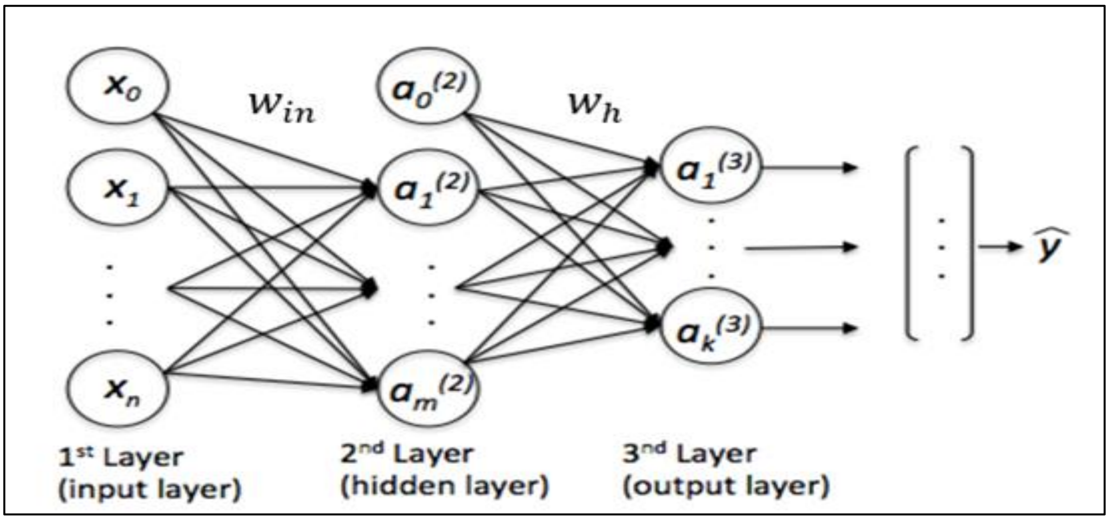


Figure (2.9): MLP Framework [63]

Each neuron in the hidden layer takes input (x_i) multiplied by a weight (w_{in}) and then activation function (f) is applied to the result (f (x_i. w_{in})). MLP contains bias neurons (a_0) in the hidden or output layers. The bias neurons are connected with all the hidden neurons or output neurons. In the above MLP Figure, a bias neuron was used with the output layer only. So, every neuron in the output layer takes the output of the hidden layer neurons (a_i) multiplied by the weights (w_h) plus the bias neuron (a_0) multiplied by the weights (w_h) and then activation function (f) is applied to the result (f (a_i. w_h+ a_0. w_h)). This has the impact of shifting the function (f) in each output node by a constant amount (a_0 * w_h)) [58].

2.10 Deep Learning for Text Classification

DL is defined as a part of ML that uses multi-layer neural networks to achieve high accuracy in many fields, such as text classification, speech recognition, image processing, etc. Based on artificial neural networks, the philosophy of DL techniques is to distribute tasks between layers, where each layer is responsible for a specific function. In addition, DL deals efficiently with unstructured data (e.g., the texts) and this is a key point that distinguishes it from traditional ML [59] (See Figure (2.10)).

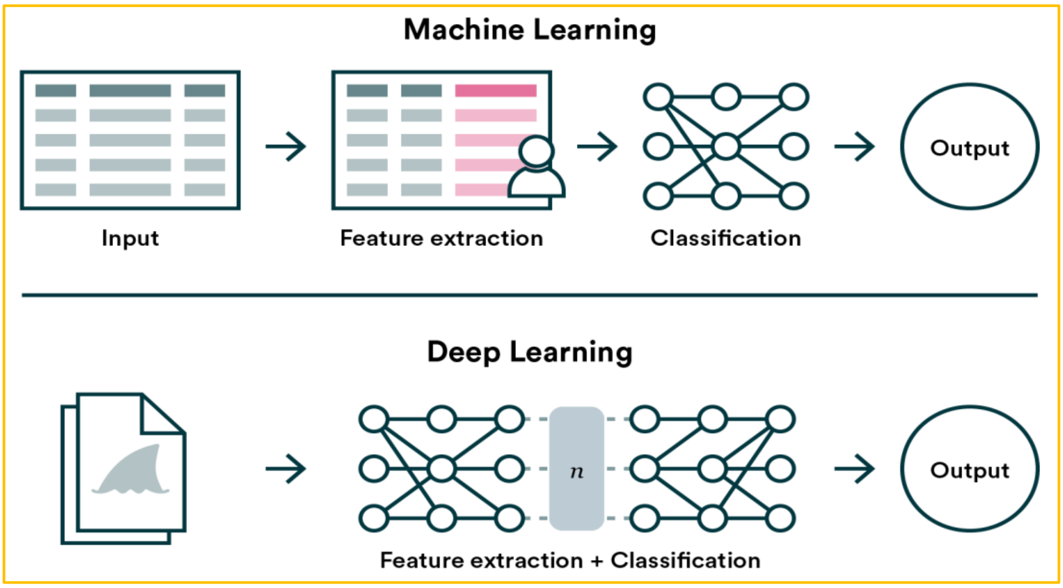


Figure (2.10): DL vs ML [59]

Recently, DL models have achieved promising results in NLP areas such as text classification. One of the main reasons behind the improved performance of DL techniques with text data is the efficient representation of words. Instead of using traditional methods such as TF-IDF to represent words, DL uses word embedding techniques that assign a vector to each word based on its semantic meaning. Among the deep learning techniques, LSTM, GRU, and CNN are the most popular text-processing techniques [60]. LSTM and GRU have achieved good performance in dealing with long and short sequences, whereas the goal of using a CNN with texts is to reduce the dimensionality of the data. MLP network was adapted with these networks for text classification [61]. More details about these four techniques are explained in the next subsections.

2.10.1 Long Short-Term Memory

LSTM network is a special traditional recurrent neural network (RNN) type. Despite the complex structure of this network, it efficiently addresses the gradient vanishing problem faced by RNN [15]. Another important benefit of LSTM is its ability to learn long-term dependencies. As shown in Figure (2.11), each LSTM neuron contains a memory cell c_t in which information can be stored. In addition, LSTM consist of three gates: input gate i_t forget gate f_t and output gate o_t. The benefit of these three gates is to regulate the information flow of the memory cell [14].

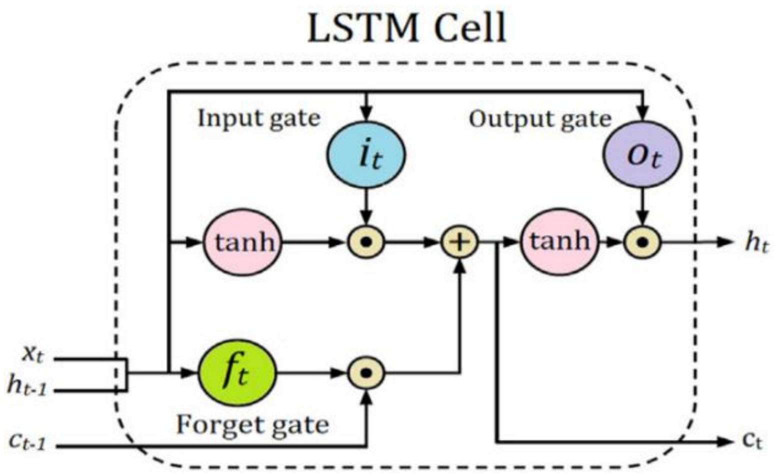


Figure (2.11): LSTM Cell [14]

Suppose tanh () and σ () are hyperbolic tangent and sigmoid functions. h_t and x_t, are the hidden state vector and input vector at time t. b is the bias vectors. U is the weight matrix for the input vector, and W is the weight matrix for the hidden state.

The forget gate focuses on the information that needs to be forgotten and is calculated according to Equation (2.14) [17].

$$f_{(t)} = \sigma\left(W_{(f)}h_{t-1} + U_{(f)}x_t + b_{(f)}\right) \qquad (2.14)$$

The input gate focuses on the information that needs to be stored and is calculated according to Equations (2.15), (2.16), and (2.17) [17].

$$i_{(t)} = \sigma\left(W_{(i)}h_{t-1} + U_{(i)}x_t + b_{(i)}\right) \qquad (2.15)$$

$$\hat{c}_{(t)} = tanh\left(W_{(i)}h_{t-1} + \ U_{(i)}x_t + b_{(i)}\right) \qquad (2.16)$$

$$c_{(t)} = f_{(t)} \odot c_{t-1} + i_{(t)} \odot \hat{c}_{(t)} \qquad (2.17)$$

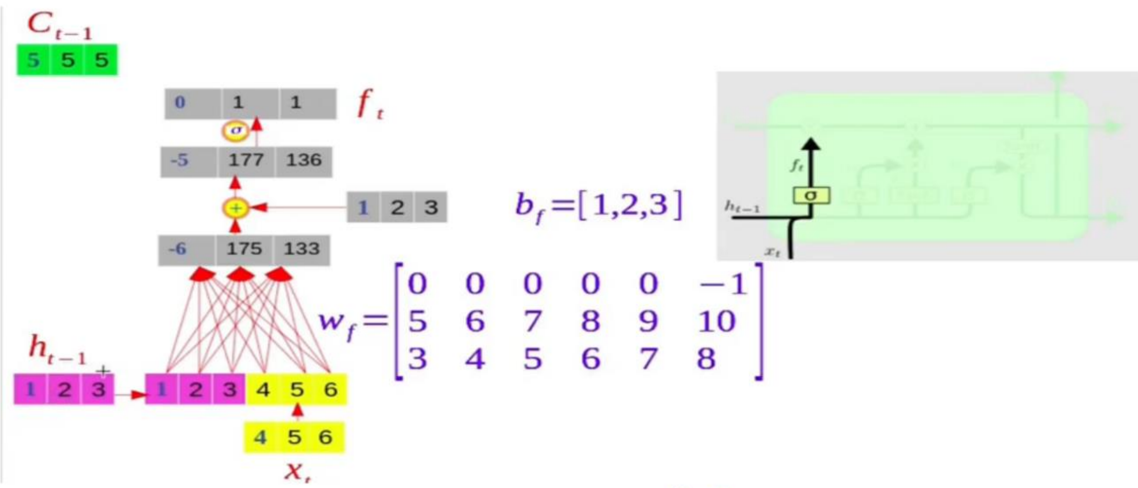
Finally, the output gate focuses on which parts need to be outputted in the cell state and is calculated according to Equations (2.18) and (2.19) [17].

$$o_{(t)} = \sigma\left(W_{(o)}h_{t-1} + \ U_{(o)}x_t + b_{(0)}\right) \qquad (2.18)$$

$$h_{(t)} = o_{(t)} \odot \tanh\left(c_{(t)}\right) \qquad (2.19)$$

The future context and preceding context are captured by combining backward ($\overleftarrow{h}_t$) and forward ($\overrightarrow{h}_t$) hidden layers. The outputs of the above equations are illustrated in the following *example*.

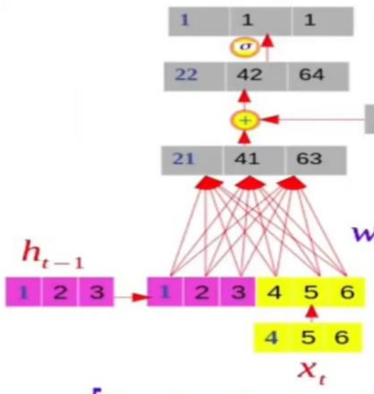
1- Forgate Gate:



$$w_f.[h_{t-1}, x_t] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix} . \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} -6 \\ 175 \\ 133 \end{bmatrix}$$

$$w_f.[h_{t-1}, x_t] + b_f = \begin{bmatrix} -6 \\ 175 \\ 133 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -5 \\ 177 \\ 136 \end{bmatrix} \quad f_t = \sigma(w_f.[h_{t-1}, x_t] + b_f) = \sigma\left(\begin{bmatrix} -5 \\ 177 \\ 136 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

2- Input Gate:



$$b_i=[1,1,1]$$

$$w_i=\begin{bmatrix}1&1&1&1&1&1\\2&2&2&2&2&2\\3&3&3&3&3&3\end{bmatrix}$$

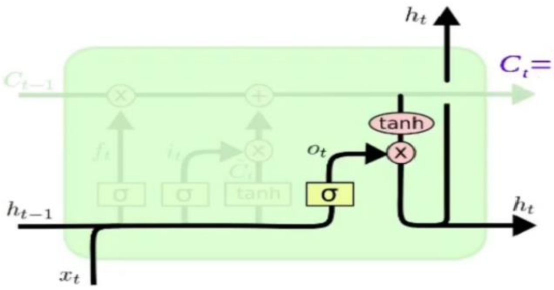
$$w_C=\begin{bmatrix}1&1&1&1&1&1\\2&2&2&2&2&2\\3&3&3&3&3&3\end{bmatrix}\quad b_C=[1,1,1]$$

$$w_C\cdot[h_{t-1},x_t]+b_C=\begin{bmatrix}1&1&1&1&1&1\\2&2&2&2&2&2\\-3&-3&-3&-3&-3&-3\end{bmatrix}\cdot\begin{bmatrix}1\\2\\3\\4\\5\\6\end{bmatrix}+\begin{bmatrix}1\\1\\1\end{bmatrix}=\begin{bmatrix}22\\42\\-62\end{bmatrix}$$

$$\widetilde{C}_t=\tanh\left(w_C\cdot[h_{t-1},x_t]+b_C\right)=\tanh\left(\begin{bmatrix}22\\42\\-62\end{bmatrix}\right)=\begin{bmatrix}1\\1\\-1\end{bmatrix}$$

$$C_t=\begin{bmatrix}0\\1\\1\end{bmatrix}*\begin{bmatrix}5\\5\\5\end{bmatrix}+\begin{bmatrix}1\\1\\1\end{bmatrix}*\begin{bmatrix}1\\1\\-1\end{bmatrix}=\begin{bmatrix}1\\6\\4\end{bmatrix}$$

3- Output Gate:



$C_t=[1\ \ 6\ \ 4]$

$$o_t=\sigma\left(W_o\ [h_{t-1},x_t]\ +\ b_o\right)$$
$$h_t=o_t*\tanh\left(C_t\right)$$

result of tanh(C_t) is [0.76,0.9999,0.9993]

Suppose O_t = [0,0.5,1]

h_t = [(0*0.76), (0.5*0.99), (1*0.99)]= [0, 0.495,0.99]

2.10.2 Gated Recurrent Unit

GRU network is a simpler version of LSTM distinguished by its capability to memorize short and long sequences. As shown in Figure (2.12), GRU has two gates: an update gate (z_t) and a reset gate (r_t). The update gate is a mix between the input gate and the forget gate that focuses on transferring information to the current state, whereas the reset gate controls whether or not the hidden state is ignored [62]. Similar to LSTM, the reset and update is calculated as follows [15]:

$$z_{(t)} = \sigma\left(W_{(z)}h_{t-1} + U_{(z)}x_t + b_{(z)}\right) \qquad (2.20)$$

$$r_{(t)} = \sigma\left(W_{(r)}h_{t-1} + U_{(r)}x_t + b_{(r)}\right) \qquad (2.21)$$

The hidden state is calculated according to Equations (2.22) and (2.23) [15].

$$\hat{h}_{(t)} = tanh\left(W_{(\hat{h}_t)}\left(h_{t-1} \odot r_{(t)}\right) + W_{\hat{h}_t}x_t\right) \qquad (2.22)$$

$$h_{(t)} = (1 - z_{(t)}) \odot h_{t-1} + z_{(t)} \odot \hat{h}_{(t)} \qquad (2.23)$$

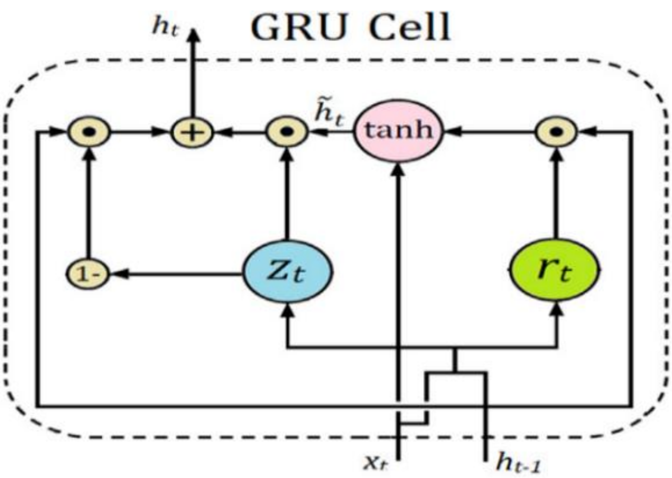


Figure (2.12): GRU Cell [15]

2.10.3 Convolution Neural Network

CNN is a deep neural network commonly used in image processing and computer vision. Recently, the researcher community has been interested in employing CNN for NLP. Text classification is achieved using CNN by applying four steps: 1) converting words into vectors through word embedding techniques; 2) identifying kernel and obtaining the local features by convolution layer; 3) selecting the most important features through max-pooling layer; 4) applying dense layer (fully connecting layer) to classify the texts or articles [63].

Figure (2.13) is an *example* of how to apply the CNN with texts for obtaining the relevant features through a convolution layer and a max pooling layer.

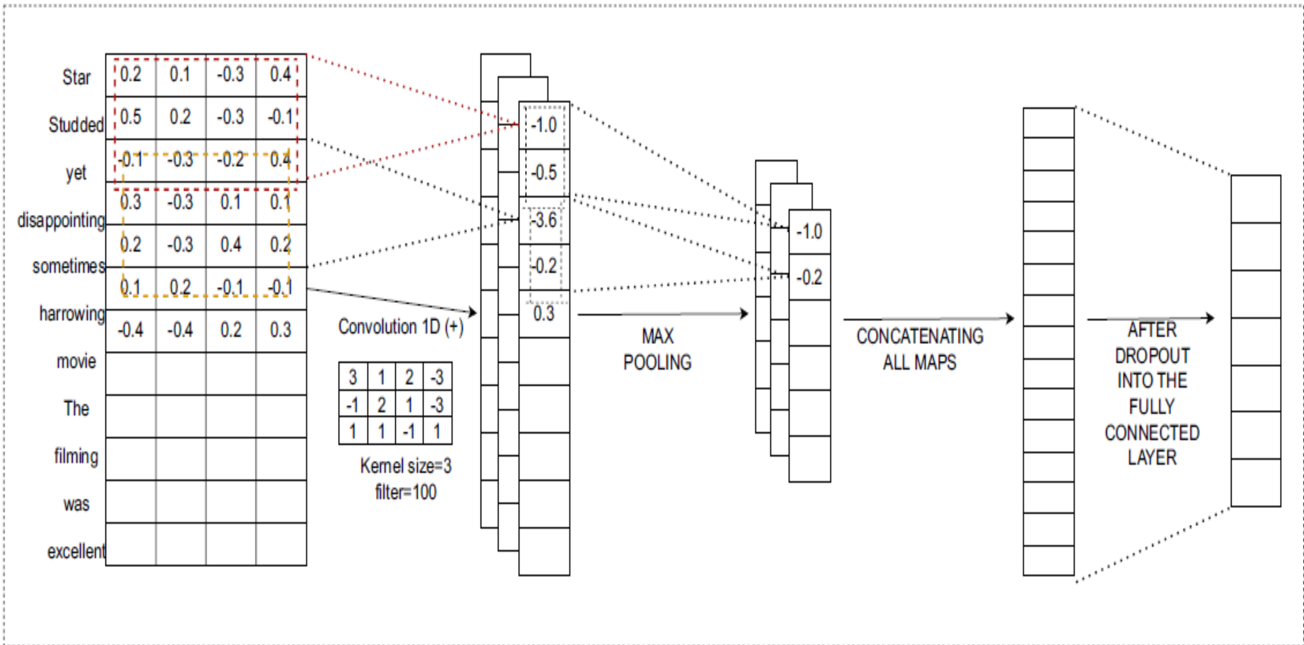


Figure (2.13): CNN Framework [63]

The convolutional is a fundamental layer of a CNN that includes most of the computation tasks. This layer consists of a set of learnable kernels (filters) by which many local features are extracted. To apply the convolution layer on a sentence (S) with (s) tokens (words), first, the vector of size (e) is generated. After that, a filter (kernel) (F) of size (e×h) is repeatedly applied to the sub-matrices of the input matrix. This, produces a local feature M = $[m_0, m_1, \ldots\ldots, m_{s-h}]$ as follows [64]:

$$m_i = F * S_{i:i+h-1} \qquad (2.24)$$

Where i = [*0, 1, …., s-h*] and $S_{i:i+h-1}$ a sub-matrix of *S* from row *i* to *i+h-1* and *h* is number of rows in the kernal. On the output of this layer, a max-pooling layer is applied to identify the relevant features. By max-pooling layer, the most important feature (b) is selected as follows [64].

$$b = \max(\mathrm{m}_i) \qquad (2.25)$$

2.11 Activation Functions

In neural networks, the activation function of a neuron is responsible for converting the summed weighted input from the neuron into the output for that input. Activation functions are mathematical operations conducted on the signal to learn linear and nonlinear tasks [65]. Linear functions are not commonly used in neural networks because they address problems with linearly separable data. However, only nonlinear activation functions allow networks to address the problem of non-linearly separable data using a small number of neurons. The most commonly nonlinear activation functions are the sigmoid activation function, hyperbolic tangent (tanh) activation function, and rectified linear unit (ReLU) activation function [66].

The sigmoid activation function is one of the most commonly used functions in neural network modelling. The function combines linear and nonlinear behaviour, transforming the entire number line into a small range between 0 and 1, as shown in Figure (2.14) [67].

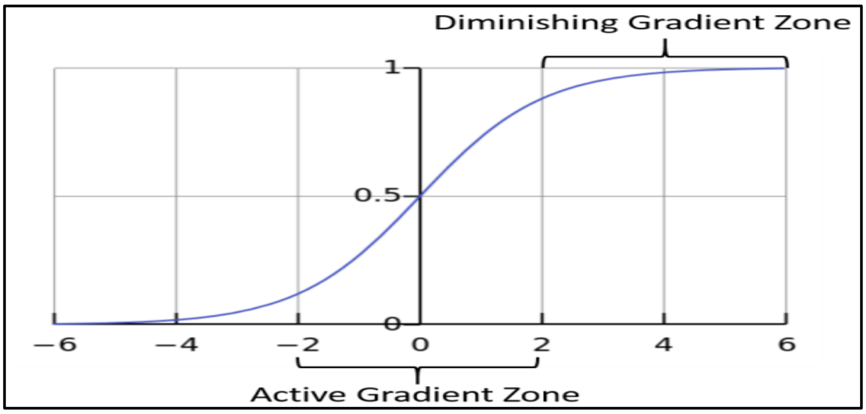


Figure (2.14): Sigmoid Activation Function [67]

The output of any neuron using the sigmoid function is interpreted as a probability, so these functions are effective in classification tasks. However, using sigmoidal functions in hidden layers is not recommended because the small gradient values lead to slow network learning. The general form of the sigmoid activation function is shown in Equation (2.13) [67].

The hyperbolic tangent activation function is another important function widely used in the neural network. Usually, hyperbolic functions are classified into three classes: sinh, cosh, and tanh. Among the three types, the tanh activation function is the most common. It is similar to sigmoid but better where its range is from (-1 to 1) rather than (0 to 1). Figure (2.15) shows the tanh activation function, while Equation (2.26) represents the general formula for this function [68].

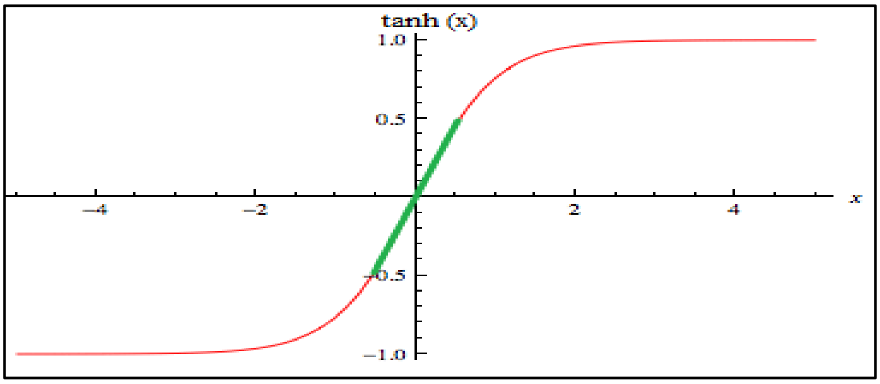


Figure (2.15): Tanh Activation Function [67]

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (2.26)$$

The sigmoid and tanh activation functions are not recommended for application in networks with many layers due to the vanishing gradient problem. The ReLU activation function, mathematically expressed in Equation (2.27), overcomes this problem, allowing neural networks to learn and train faster and perform better. ReLU is the default activation function when developing MLP and CNN. As shown in Figure (2.16), ReLU is a nonlinear function that outputs the same input directly if it is positive; otherwise, it output zero [67].

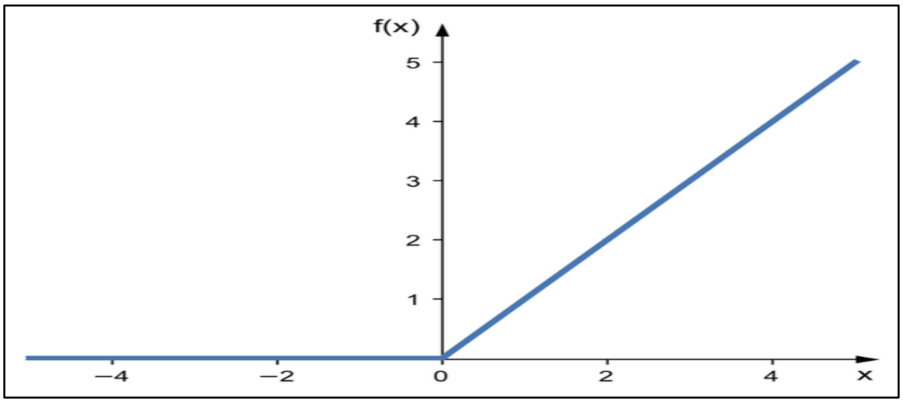


Figure (2.16): ReLU Activation Function [67]

$$ReLU\ (x) = \max(0, x) \qquad (2.27)$$

2.12 Forward Propagation

Forward propagation moves from the input layer (the first layer) to the output layer (the last layer) in a neural network. In forward propagation, the input data is calculated, stored, and then fed in the forward direction to generate the desired output [68]. At each neuron, the processing is performed in two steps: *preactivation* and *activation*. The weighted sum of inputs is calculated before applying the activation function in preactivation, as shown in Equation (2.28) [69].

$$z = w.x + b \qquad (2.28)$$

Where: w is the weight vector, x represents the input vector, and b is the bias. At first, weights and biases parameters are initialized randomly. When the activation function (f) is applied to the resultant value of the preactivation step, this state is called activation. The hidden activation vector was obtained by this step, as shown in Equation (2.29) [69].

$$h = f\ (z) \qquad (2.29)$$

The output of the neural network was obtained by the output layer. In this layer, the sigmoid activation function is usually applied. In general, the output of any node in the neural network was obtained by the general Equation (2.30) [69].

$$h^l = f\ (w^l .\ h^{l-1} + b^l) \qquad (2.30)$$

In forward propagation, the performance of a neural network is improved by defining some necessary parameters. Instead of feeding the neural network all the examples in the dataset at once, the *batch size* was proposed to control the number of examples that will be passed to the neural network at one time. The goal of dividing the dataset into batches is to facilitate the computational operations and thus improve the model's performance. At the end of the batch, the expected values are compared with the actual values in order to compute the loss function [69]. Once the loss function is calculated, the backpropagation stage begins. In general, three common techniques are dealing with the neural network training mechanism: (1) when the batch size is more than one sample and less than the training set size, this technique called *mini-batch gradient descent*; (2) when the batch size equal to training set size, this technique called *batch gradient descent; (3)* when the batch size equal to 1, this technique called *stochastic gradient descent* [70].

The loss function or so-called cost function is an important criterion for how well a neural network model is in a prediction task. As shown in Equations (2.31) and (2,32), the loss function is the absolute difference between the actual and expected values of a particular instance in the dataset. In contrast, the cost function is the sum of the difference between the actual and expected values of all instances in the dataset divided by the number of instances [71].

$$loss = abs\ (predected - Actual) \qquad (2.31)$$

$$cost = \frac{\sum_{i=1}^{n} loss_i}{n} \qquad (2.32)$$

Therefore, the cost value defines the neural network's error, and the main goal is to minimize it. The loss function depends mainly on network parameters (biases and weights). So, this function was reduced by updating these parameters through learning or optimization methods (see backpropagation section). There are many types of loss functions. The most common are: (1) mean square error loss with regression task; (2) cross-entropy with a multi-classification task; (3) binary cross-entropy with binary classification task [72].

With binary classification problems, binary cross-entropy is applied as a loss function to compute the difference between predicted and actual values. Binary cross-entropy can be expressed as follow [72]:

$$BCE\ (y_i, \hat{y}_i) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \hat{y}_i) + (1 - y_i)(1 - \log(\hat{y}_i)) \qquad (2.33)$$

2.13 Backpropagation

Unlike forward propagation, backpropagation always starts from the output layer (the last layer) and propagates backward to the input layer. Backpropagation is the heart of the neural network, as its responsibility is to update the weights and biases for each layer. The desired output can be obtained from the neural network by adjusting these weights and biases. These parameters are updated by calculating the gradients efficiently [73].

To update the weights and biases, it is first necessary to calculate the cost function of the neural network (see Figure (2.17)). Then, the chain rule is used to calculate the derivative of this function with respect to weights and biases. Finally, gradient (derivative) is applied to obtain new weights and biases. This procedure is applied to each neuron in the neural network to update its weights and biases [74].

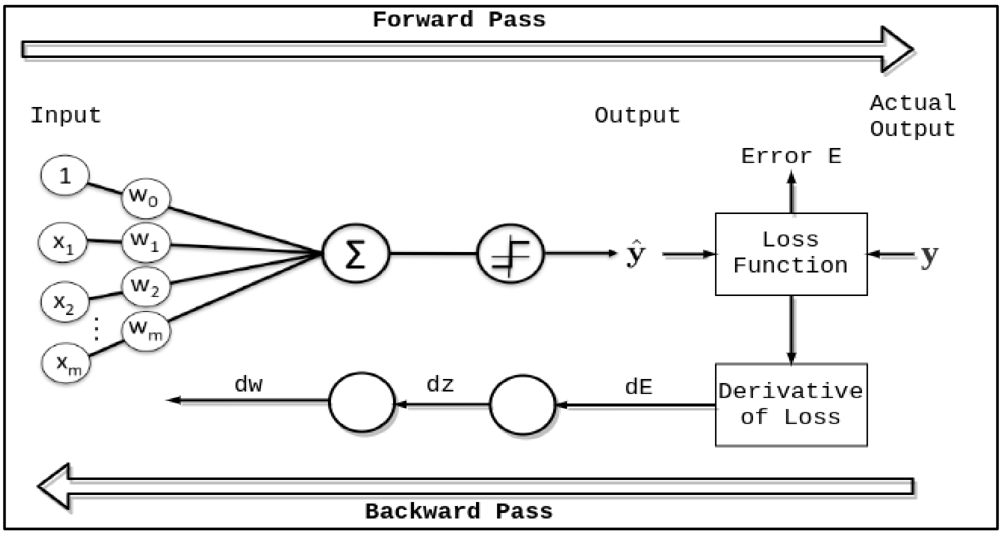


Figure (2.17): Forward and Backward Propagation [74]

Suppose that ($h = f\ (z)$) is the output of any neuron in the network with cost function (C). Updating the weights (w) and biases (b) of this neuron is given by [74]:

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial h} . \frac{\partial h}{\partial z} . \frac{\partial z}{\partial w} \tag{2.34}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial h} . \frac{\partial h}{\partial z} . \frac{\partial z}{\partial b} \tag{2.35}$$

The gradient descent (or any optimization algorithm) is then applied to obtain new weights and biases as follows [74]:

$$\widehat{w} = w - \alpha . \frac{\partial C}{\partial w} \tag{2.36}$$

$$\hat{b} = b - \alpha . \frac{\partial C}{\partial b} \tag{2.37}$$

Where: $\boldsymbol{\alpha}$ is a learning rate. The learning rate is an important hyper-parameter utilized to control the rate at which a neural network updates the values of the weights or biases at each epoch (epoch means one complete pass of all instances in the training set through the neural network). As shown in Figure (2.18), the challenge lies in choosing the ideal value for the learning rate because a value that is too small leads to a slowdown in the training of the neural network, while a value that is too large produces non-ideal weight [75].

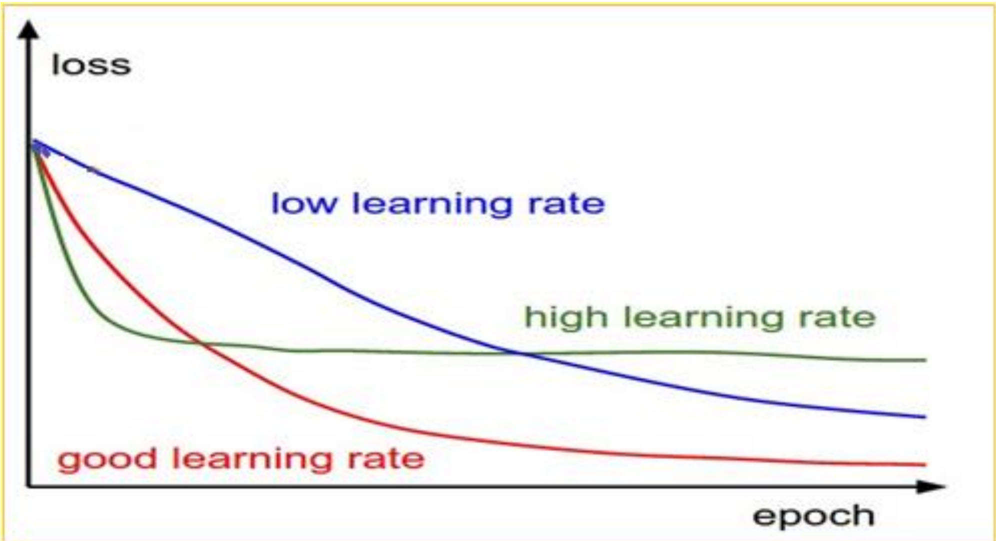


Figure (2.18): Learning Rate with Different Values [75]

2.14 Neural Network Optimization

In the backpropagation phase, several optimizers are used for training the neural network. While training the neural network using optimizers, the parameters are modified to minimize the cost function. An optimizer is an algorithm or function designed to adjust the parameters of the neural network, increasing the model's efficiency and accuracy. The most important optimizers for updating neural network parameters are root mean squared propagation (RMSProp), stochastic gradient descent (SGD), adaptive moment estimation (Adam), Adaptive Gradient (AdaGrad), and others. Adam's optimizer is the most widely used and recommended by many researchers [76]. This optimizer is an extension of the gradient descent technique and a natural successor to SGD and RMSProp techniques that efficiently update network parameters [77]. Adam is a perfect algorithm that combines the benefits of both AdaGrad and RMSprop models. Like AdaGrad, Adam is an adaptive learning rate technique that uses the moving average of the gradient instead of the gradient itself. In addition, Adam focuses on the properties of the RMSprop by using the squared gradients to scale the learning rate [78].

Adam optimizer differs from other optimization algorithms by using estimations of the gradient's first and second moments (the momentum term) to adapt the learning rate parameter for each weight and bias of the deep neural network. The first moment is mean (m_t), and the second moment is uncentered variance (v_t) (this means that the mean is not subtracted by calculating the variance). If (g_t) represents a gradient, and β_1 and β_2 represent two parameters for tuning, the neural network is trained by updating the network's weights and biases as follows [79]:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.38}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \tag{2.39}$$

To correct the bias and obtain the desired values for the first and second moments, the Adam optimizer adds an important step: bias correction. The main reason to include this step is that somehow it prevents the bias of the initialization of (m_t) and (v_t) to zero. The new values of first and second moments are obtained as follows [79]:

$$\widehat{m}_t = \frac{m_t}{1 - {\beta_1}^t} \tag{2.40}$$

$$\hat{v}_t = \frac{v_t}{1 - {\beta_2}^t} \tag{2.41}$$

Finally, the weights and biases are updated, which in turn train the neural network through the following [79]:

$$w_t = w_{t-1} - \propto \frac{\widehat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.42}$$

Where: ε is a small value that added to avoid division by zero. The recommended values of Adam parameters are as follows: $\epsilon = 10^{-8}, \beta_1 = 0.9, \beta_1 = 0.999, \propto = 0.001.$

2.15 Neural Network Problems and Solutions

The deep neural network become one of the main research areas in developing and improving intelligent machines. Deep Learning algorithms emulate human brains by neurons and progressively learn and train to accurately solve any problem. Despite the benefits of deep neural networks, there are many challenges in building a deep model architecture. There are many challenges when building a deep neural network model which are overfitting, underfitting, gradient vanishing, overtraining, and slow training. The research community contributed to providing appropriate solutions for each of these challenges [80].

Overfitting and underfitting are mainly responsible for the poor performance of deep learning algorithms. Two important terms define the presence or absence of overfitting or underfitting in a neural network model: bias and variance [81]. The bias is simply the error rate of the training set, whereas the variance is the difference between the error rate of the training set and the test set. Underfitting occurs when the model has a high bias in the training set. Thus, the model cannot perform well on the training set and cannot generalize to new data. On the other hand, overfitting occurs when the deep model has a high variance and thus it performs well on the training set but does not work correctly on the testing set. For more details, see Figure (2.19) [82].

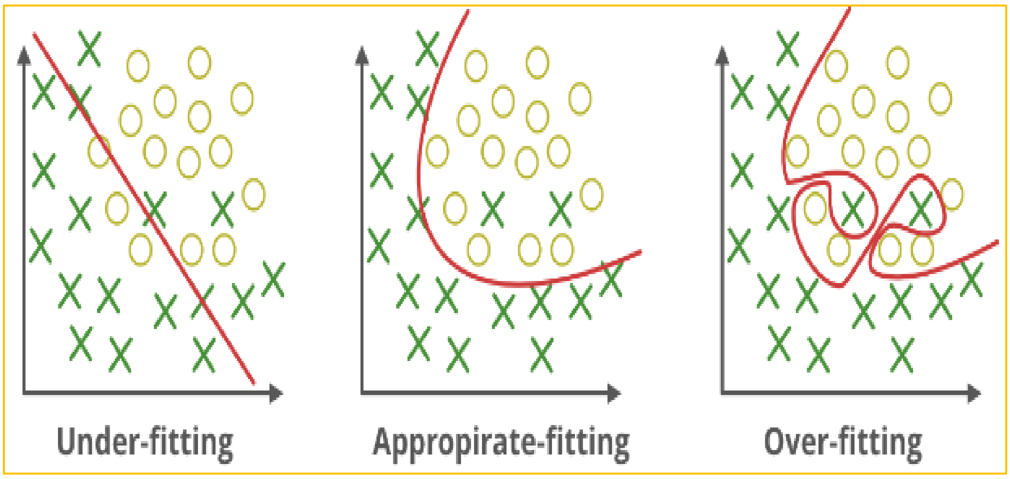


Figure (2.19): Deep Model Fitting [82]

2.15.1 Dropout

In deep neural networks, there are different architectures, sometimes simple and sometimes complex, to generalize the dataset. Usually, deep neural networks quickly overfit a training set with few instances. In contrast, neural networks with different model configurations reduce overfitting but need the additional computational expense of training. The dropout mechanism is the simple and common method to process overfitting in a deep neural network [83].

The term "dropout" in deep neural networks refers to dropping out the neurons in a neural network to create a new deep learning architecture out of the parent deep neural network (as shown in Figure (2.20)). The dropping-out process for neurons is performed randomly during training. Dropout offers a remarkably effective technique to reduce overfitting and optimize generalization error in deep neural networks of all types. In overfitting, complex co-adaptations between neurons are generated through sharing errors in successive layers. Therefore, the dropout prevents this complex co-adaptation by deleting the nodes that spread errors with a certain probability [84].

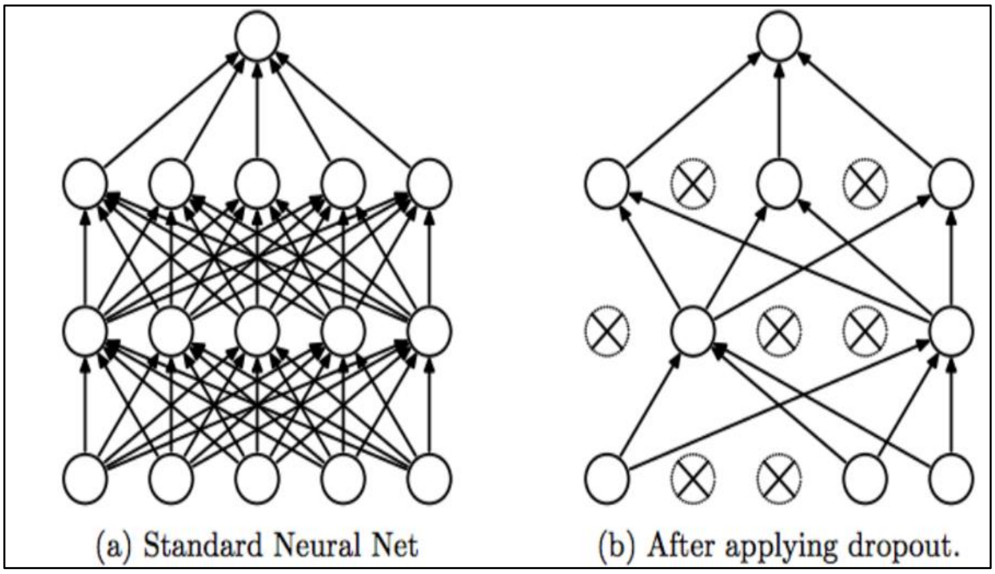


Figure (2.20): The Concept of Dropout [84]

Dropout is implemented with most types of layers, such as fully connected layers, recurrent layers, convolutional layers, etc. With dropout, a new hyperparameter is provided, which specifies the probability at which outputs of the layer are dropped out (or the probability at which outputs of the layer are retained). Each node remains with a fixed probability (p) independent of others, where (p) can be chosen using a validation set. Half of the network nodes are dropped when the probability (p) is set to 0.5 [85].

2.15.2 Early Stopping

The main challenge in training a deep neural network is how long to train it. Too little training means the model may underfit the training and testing sets. Too much training means that the model may overfit the training set and perform poorly on the testing set. As shown in Figure (2.21), the solution is to train the model on the training set but stopping the training at the point when performance on a validation set starts to degrade. Early Stopping is an effective and widely utilized approach for training deep neural networks [86].

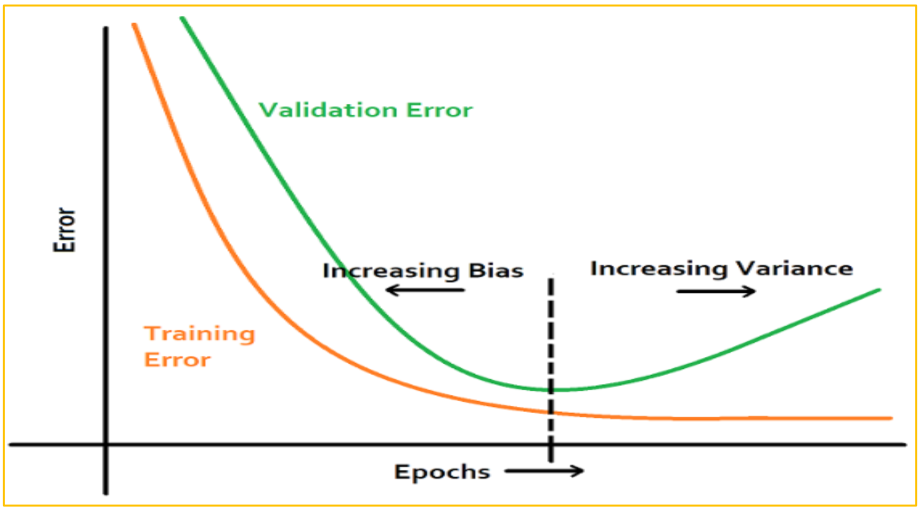


Figure (2.21): Early Stopping [86]

The problem of overtraining was addressed through many approaches. One method to solve this problem is using the number of training epochs as a hyperparameter and then training the deep neural network model multiple times with different values. Then, the number of epochs is selected based on the best performance of the deep neural network. The downside of this method is that it is computationally ineffective and takes a long time, especially for large deep-learning models trained on large data over hours or days [87].

An alternative approach is early stopping which trains the deep neural model once for a large number of epochs. The deep model is evaluated on a validation set after each epoch during training. If the performance of the deep model begins to degrade (e.g., loss starts to increase or accuracy starts to decrease), then the training is stopped [88].

Early stopping depends on an important hyperparameter called patience. Suppose this hyperparameter is set to (p) with a number of training epochs or training iterations. In that case, the training process will not be stopped only if there is no improvement in the performance deep neural network model for (p) iterations or epochs in a row [88].

2.15.3 Batch Normalization

Normalization (at large) is a pre-processing step used to transform the numerical values to a specific scale without distorting its shape. Batch normalization, or so-called batch norm, is a technique utilized to train deep neural network algorithms faster. The term "Batch" in batch normalization means that the normalizing process is performed in the batches of epochs, not as a single input [89].

Training deep learning algorithms with many layers is challenging where they are sensitive to the configuration of the learning model and the initial random weights [90]. The reason for this difficulty is the distribution of the inputs for each layer may change after each batch when the weights are updated. This causes the problem of internal covariate shift and makes the deep neural network model slower and unstable. Thus, batch normalization is proposed to address these problems [91].

Batch normalization aims to ensure that the input for each layer is distributed with the same mean and standard deviation (see Figure (2.22)). By normalizing the inputs, the deep neural network model training dramatically reduces the number of epochs needed. It also makes the model faster and more stable [91].

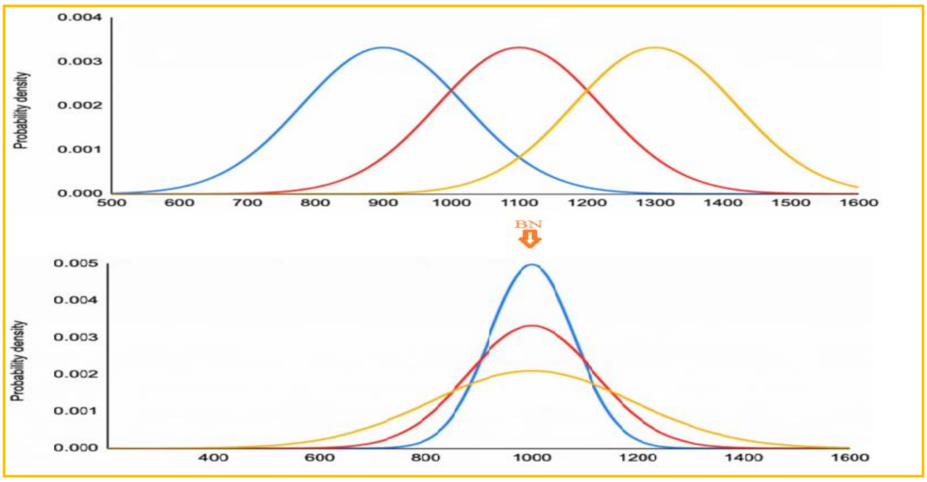


Figure (2.22): Batch Normalization [91]

To apply batch normalization on vectors (h_i) over a minibatch B of size m ($B = \{h_1, h_2, \dots, h_m\}$), it is first necessary to calculate the mean (μ) as follows [92]:

$$\mu = \frac{1}{m}\sum\nolimits_{i=1}^{m} h_i \tag{2.43}$$

Once the mean is calculated, the next step is to calculate the standard deviation (σ) of the vector (h) as follows [92]:

$$\sigma = \frac{1}{m}\sum\nolimits_{i=1}^{m} (h_i - \mu)^2 \tag{2.44}$$

The hidden activations are normalized using the mean, the standard deviation, and the smoothing term (ϵ) that prevents a division by a zero value as follows [92]:

$$h_{i(norm)} = \frac{h_i - \mu}{\sigma - \epsilon} \tag{2.45}$$

2.16 Evaluation Measures

The performance of an ML or DL model is measured through data division strategies and performance metrics. Therefore, this section discusses two important strategies for dividing the dataset into a testing set and a training set, as well as clarifies the most important metrics used to evaluate the performance of the classification model.

2.16.1 Train-Test Splitting Strategies

After the data pre-processing and feature selection but before beginning to train ML or DL models, it is important to have a strategy for dividing the dataset into training and testing sets. Two common strategies are used to achieve this purpose: *holdout* and *cross-validation* [93].

As shown in Figure (2.23), the holdout technique for training the prediction models is used to split the dataset into two sets: one for training and the other for testing. The prediction model is trained on the training set. The testing set is then used to test the model and to know how well it predicts new instances. The original dataset is usually divided into (70%) for training and (30%) for testing [94].

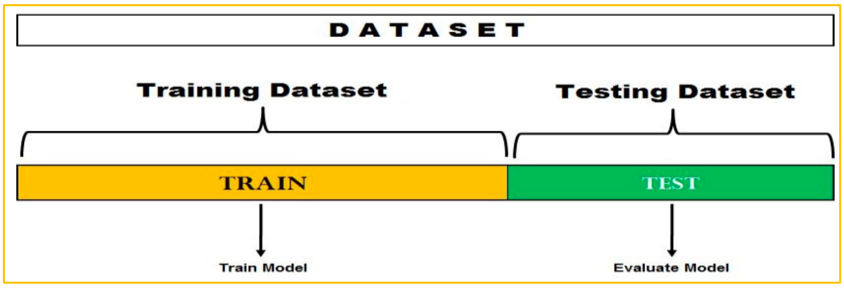


Figure (2.23): Holdout Method [94]

K-folds cross-validation is common and widely used for training and testing a prediction model. In this approach, the data is partitioned into (k) equal parts, as shown in Figure (2.24). Using the data, (k) times are performed for training and testing the model. Each (k) part is utilized in one iteration as the testing set and in the other (k-1) iterations as part of the training set. In the end, the performance metric is used in all iterations to determine an average error rate or accuracy [95].

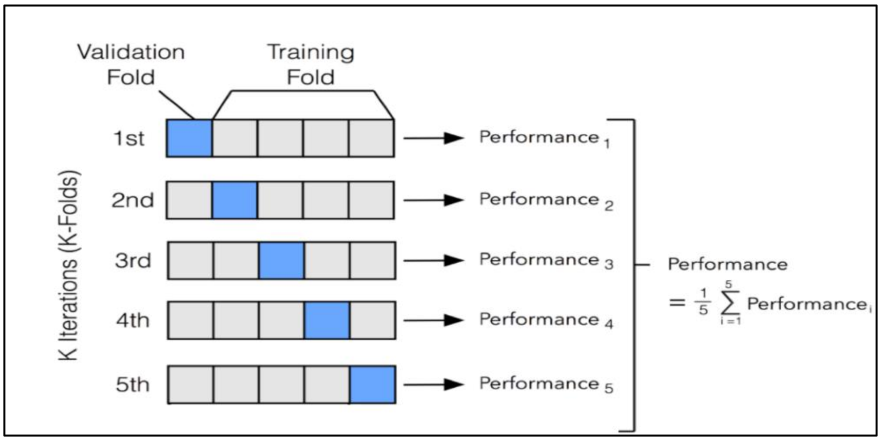


Figure (2.24): Cross-Validation Method [95]

2.16.2 Performance Metrics

Evaluating a DL or ML model's performance is one of the main steps while building an effective model. Different metrics are utilized to evaluate the quality of any model, which are called evaluation metrics or performance metrics. These evaluation metrics help to understand how well the prediction models have performed for the given dataset. In addition, the evaluation metrics are used to improve the model's performance by tuning the parameters [96].

In DL or ML, each task is divided into Regression or Classification. Not all metrics are utilized for all types of tasks; thus, it is important to understand which metrics should be utilized. Four important metrics are used for the classification task: accuracy, precision, recall, and f-measure [97].

On the other hand, the confusion matrix is a performance measurement for a prediction model where the output can be two or more classes. As shown in Table (2.1), the confusion matrix is a table with 4 different combinations of actual and predicted values that is extremely useful for measuring precision, recall, f-measure, and accuracy [98].

Table (2.1): Confusion Matrix

		Predicted Class	
		+	-
Actual Class	+	*TP*	*FN*
	-	*FP*	*TN*

Where TP represents True Positive, the number of records (instances) from class positive predicted correctly as class positive. TN represents True Negative, the number of instances from class negative predicted correctly as class negative. FP represents False Positive, the number of instances from class negative predicted incorrectly as class positive. FN represents False Negative, the number of instances from class positive predicted incorrectly as class negative [98].

Accuracy is the number of correctly classified examples over the total number of data examples (see Equation (2.45)). Accuracy is a good measure if the dataset is balanced (this means that the number of positive and negative examples is equal in number or close) [99].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.45}$$

Precision is a metric that identifies the number of correct positive predictions made. As shown in Equation (2.46), precision is computed as the ratio of correctly predicted positive instances divided by the total number of positive instances [99].

$$Precision = \frac{TP}{TP + FP} \tag{2.46}$$

Unlike precision which only indicates the correct positive predictions out of all positive predictions, recall indicates missed positive predictions. As in Equation (2.47), Recall can be calculated by dividing true positive predictions by true positive predictions plus false negative predictions [99].

$$Recall = \frac{TP}{TP + FN} \tag{2.47}$$

F-Measure introduces a method to combine precision and recall into one measure that captures both properties. Precision alone or recall alone does not give a complete perception of the model, as the model can achieve excellent precision but poor recall or vice versa. So, f-measure provides a way to express both metrics to one score. The f-measure is computed as follows [99]:

$$F - Measure = \frac{2(Precision * Recall)}{Precision + Recall} \tag{2.48}$$

Chapter Three

The Proposed System

The Proposed System

3.1 Overview

This dissertation introduces the DMLM, ABHDM, and EDLM for improving the prediction of fake news. It focuses on long news articles circulated in news agencies and short articles or tweets exchanged on social media.

3.2 The Proposed System Architecture

As shown in Figure (3.1) the proposed system uses traditional machine learning and deep neural networks for FND. The proposed system views Three developed models: DMLM, ABHDM, and EDLM. The DMLM is a developed model for traditional machine learning methods, whereas the ABHDM and EDLM are two new and improved architectures of deep neural networks.

The DMLM is concerned with detecting Arabic fake news circulating on social media. This model focuses on the user-based features and contained-based features on the one hand, and the features extracted from the text (tweet) on the other hand. The proposed model improved its performance by constructing a FM for selecting important user-based features. The FM selects relevant features of the user based on two proposed models for feature selection: MDGI and DFI.

On the other hand, a ABHDM is proposed for addressing arbitrary and shorter news article. This model focuses on the most informative word in the classifying any news article by relying on the attention mechanism. The EDLM is another deep model that makes its decisions based on training three deep neural networks: CNN, Bi-LSTM, and BiGRU. Each network is independently trained and then used to classify unseen examples.

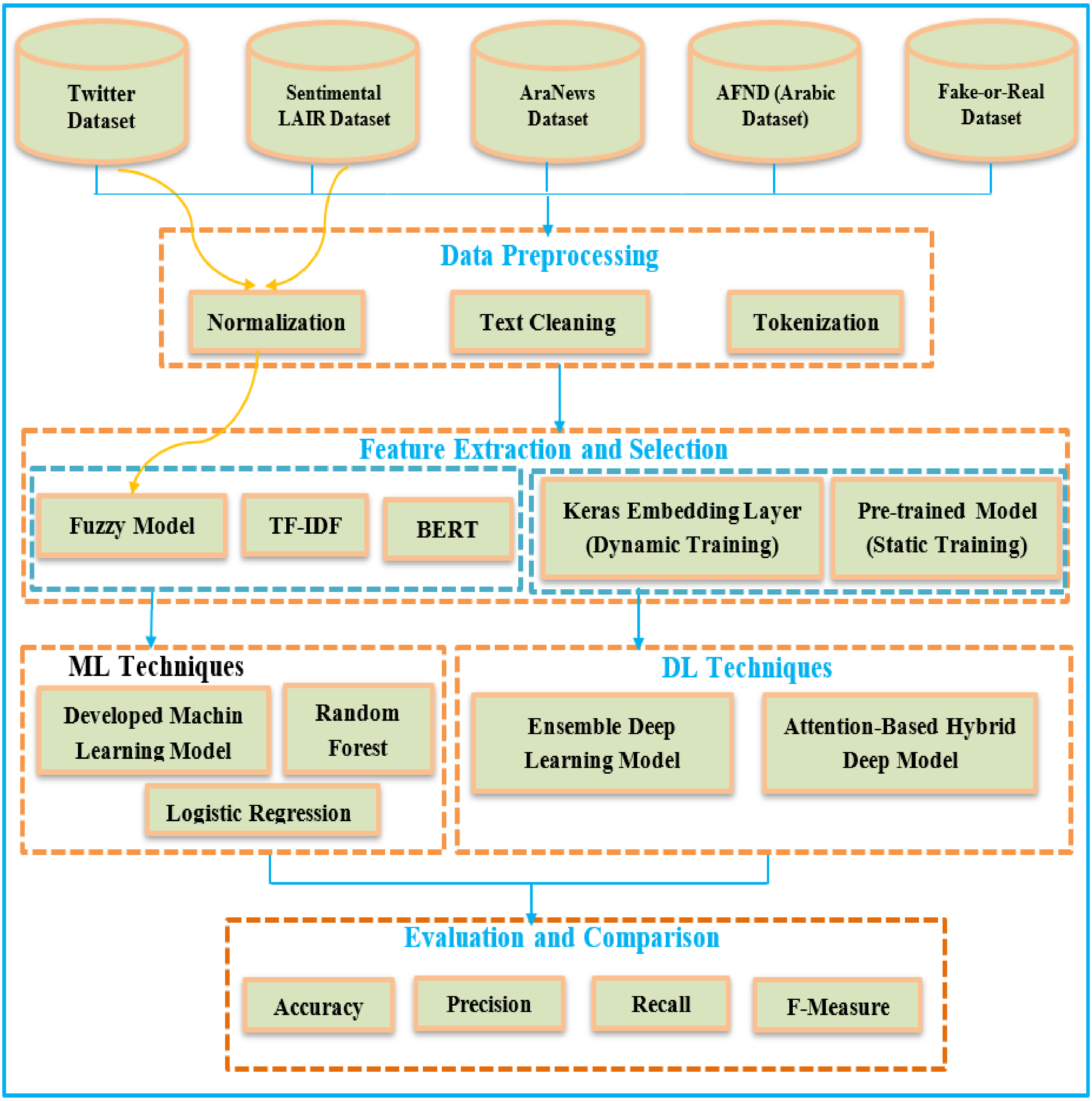


Figure 3.1: The Proposed System Framework

All three models depicted by the proposed system include four stages to achieve their objectives: the first stage is data preprocessing which includes normalization and text cleaning and processing. The second stage is feature extraction and feature selection. Traditional feature extraction methods such as TF-IDF were used in extracting features, and modern feature extraction methods represented by word embedding methods. On the other hand, fuzzy logic was employed to develop a feature selection method for identifying relevant user-based features. The third stage includes building prediction models. At this point, the random forest classifier was developed and two new architectures for deep neural networks were proposed. Finally, the fourth stage is the evaluation of the performance of developed models based on common evaluation metrics such as accuracy, precision, recall, and f-measure.

3.3 Data Pre-processing Stage

Data preprocessing is a necessary procedure in building prediction models because the raw tweets or news articles are not in the optimal format to be fed into these models. The preprocessing in this dissertation has been achieved in three basic steps: (1) data cleaning; (2) deleting meaningless news articles and short news articles; and (3) tokenization.

The datasets used in this dissertation (especially Twitter dataset) contain many unwanted words or noises. The data cleaning step focuses on deleting unnecessary information (noises). Unnecessary information removed in this step are symbols, punctuation, URL, username, and stop words. Finally, Arabic stemming is performed to reduce the maximum number of words. In this dissertation, Arabic stop words and Arabic stemming were used with machine learning models (i.e., they were not applied with deep learning models). Table (3.1) explains some Arabic stop words, whereas Table (3.2) shows some examples of Arabic stemming.

Table (3.1): Some of the Arabic Stop Words

Arabic Stop Words	Arabic Stop Words	Arabic Stop Words
ليست	عند	لكن
كما	عندما	متى
فلا	على	مهما
أي	بين	ومن
ان	له	هذا
او	فإذا	أنتما
أي	أنت	كم
أكثر	به	فيها
كيفما	لكما	كيف
حيث	كلتا	إليكما
الى	وإذا	به

Table (3.2): Stemming for Some Arabic Words

Arabic Word	Arabic Stemming
السياسة	سياسة
الاقتصادي	اقتصاد
كرة القدم	كرة قدم
فايروسات	فايروس
المريض المصاب	مريض مصاب
التحالف الدولي	تحالف دولي
and so on	

As shown in Table (3.3), the AraNews dataset and AFND contain many short articles that don't make sense. Also, some articles in these two datasets cannot be considered news. Such articles may mislead the prediction model. Therefore, data pre-processing included the deletion of all news articles containing ten words or less and the deletion of all news articles with no meaning. Finally, tokenization is conducted by dividing the text (news articles or tweets) into words based on spaces between them.

Table (3.3): Meaningless News Articles

Description	News Article
It's Short News	قد تمر البلاد بأزمة مجاعة
It's Meaningless News	اللهم احفظ الشعوب العربية والاسلامية من التدخلات الخارجية وانصر دولة فلسطين

For user-based features in the Twitter dataset and Sentimental LIAR dataset, the values of all features have been normalized between zero and one based on the min-max normalization.

3.4 Feature Extraction Stage

Converting a particular news article or tweet into vectors is a major task in text processing. This principle falls within the concept of feature extraction that focuses on transforming text which prediction models do not accept into applicable text. Many traditional techniques are available for feature extraction, the most important of which is the TF-IDF technique. On the other hand, the modern trend is to use pre-trained models for word representation. This dissertation used TF-IDF technique and three pre-trained models for word representation: BERT, GloVe, and FastText.

Each technique was applied to represent a particular dataset from the five datasets used in this dissertation. The TF-IDF is used to convert the Arabic words in Twitter dataset into features. BERT pre-trained model is applied on Sentimental LIAR dataset. The GloVe pre-trained model is exploit to represent the words in Fake-or-Real dataset. Finally, matrix of vectors was built for each of the AraNews dataset and the AFND, depending on each of FastText and GloVe pre-trained models.

The TF-IDF technique focuses on the number of times that the term (word) appears during all tweets in the dataset. Three main steps represent the words in each tweet. First, a list of unique words (vocabularies) from all tweets is created. From all vocabularies, a list of features has been created in which each feature consists of a pair (key, token) where the key represents the index in the list and the token refers to the feature. Second, the tweets have been represented as a vector space of features where each tweet is an array of features. The number of columns corresponds to the total number of unique words (features) for the Twitter dataset as shown in Figure (3.2). Finally, each feature can be weighted using TF-IDF according to equation (2.2).

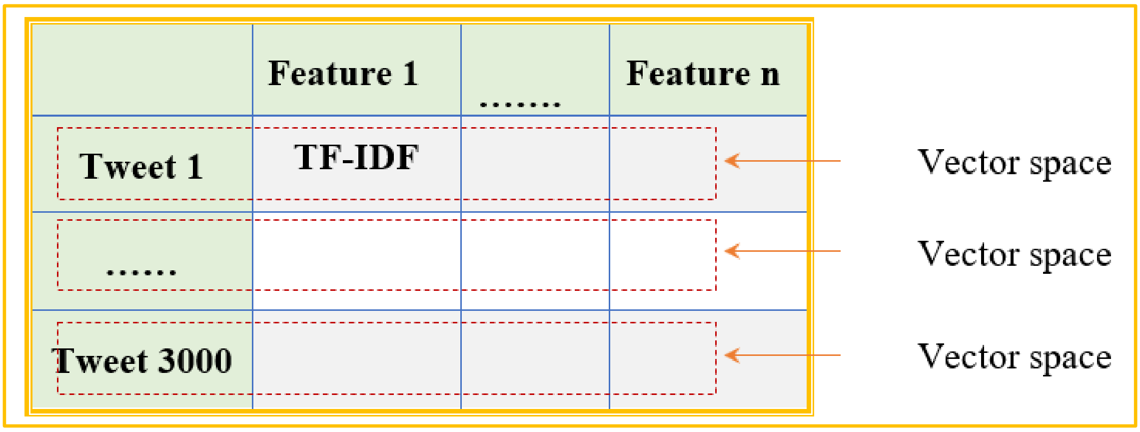


Figure (3.2): TF-IDF for Feature Extraction

The BERT pre-trained model was used to convert every sentence in the Sentimental LIAR dataset into a vector containing embedding values for the entire sentence, not for each word. Thus, we use this model to generate embedding values for 12786 sentences in the Sentimental LIAR dataset. The output of the BERT pre-trained model in each hidden state is a vector of size 768 where the vector corresponds to the CLS token. For all sentences in the Sentimental LIAR dataset, the output of BERT is an embedding matrix of size (768*12786). BERT Architecture for word representation was explained in Figure (3.3).

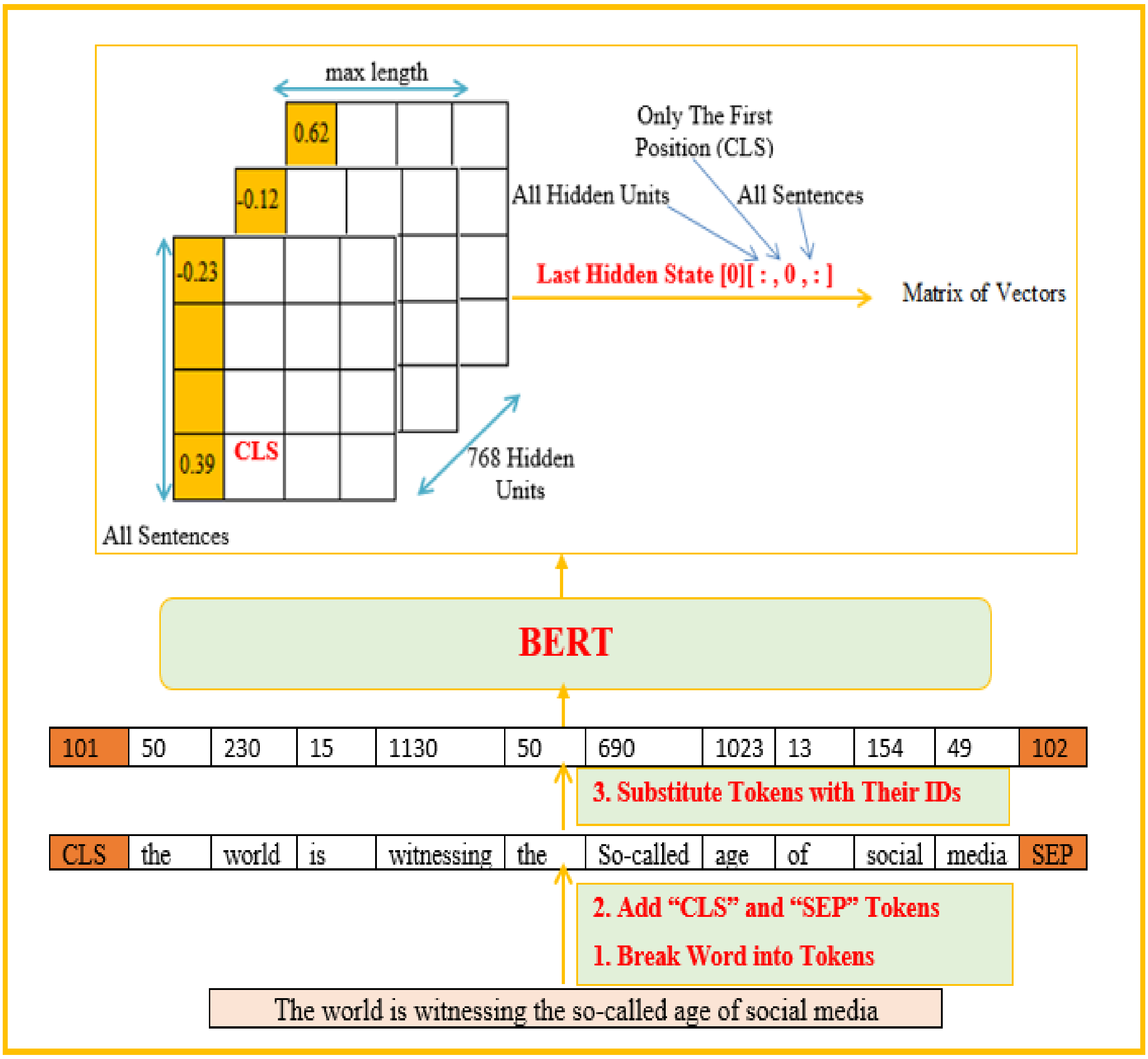


Figure (3.3): BERT Architecture for Word Representation

Two pre-trained models have been used to build the input matrix for the AraNews dataset and AFND: FastText and GloVe. The AFND embedding matrix and AraNews embedding matrix are built based on these models. These matrixes represent the initial weights of the embedding layer. The AFND embedding matrix contains the embedding values for the words in the AFND. Likewise, The AraNews embedding matrix contains the embedding values for the words in the AraNews dataset. FastText is mainly used for word representation. In contrast, the GloVe is used to represent words that do not have vectors in FastText. In addition, any word that was not represented in both models is assigned a vector based on the closest word to it. Two main benefits behind this matrix: (1) increase the efficiency of the model by searching in a small matrix; (2) increase accuracy by addressing the "out of vocabulary" problem.

It should be noted that any embedding matrix is constructed depending on three parameters: max words (top_words), embedding dimension (emb_dim), and max length (max_len). These parameters are chosen based on the nature and size of the datasets. The top_words parameter is assigned based on the number of unique words available in the dataset. The max_len is determined by the length of each news article in the dataset. Depending on this parameter, the padding or truncating technique is used. The emb_dim parameter defines the number of embedding values in each vector.

Algorithm (3.1) describes the data pre-processing steps as well as the procedures taken in the feature extraction process.

Algorithm 3.1: Data Preprocessing and Feature Extraction

Input: *Dataset of news articles*

Output: *Two Arrays (Pre-processed Dataset[N*M]) where N is number of rows (vectors) and M is number of columns (number of embedding values in each vector)*

Begin

1. *Read Dataset*

// Data Cleaning

2. ***for*** *i = 1 to n* ***do*** *//n is a number of news articles*
3. *removing punctuations, symbols, username, URL, and stop words*
4. ***end for***

// Removing Short News Articles

5. ***for*** *i = 1 to n* ***do***
6. ***if*** *len(i) <= 10* ***then***
7. *drop news article (i) from dataset*
8. *end if*

```
    end for
                        // Tokenization
    for Term t in Dataset do
        extract tokens Ts of news article based on space between words
        SOT[t]= Ts              // Set of tokens (SOT)
    end for
    Set max unique words (top_words), embedding dimension (emb_dim),
    and max length for news article (max_len)
                // Feature Extraction
    MOV is a matrix of vectors for dataset.
                // TF-IDF Matrix
    Finding unique words in dataset and listing them as features
    for every tweet in dataset do
          Converting the tweets' texts into sparse feature vectors
          Calculating the weight of each feature using Equation (2.2)
          Saving the results in MOV
    end for
                // Word Embedding Matrix
    GLV is GloVe words embedding and FT is FastText words embedding
    for each word (w) in SOT do
        if word vector (WV) of w exist in FT
             EM [w] = WV        // EM is embedding matrix
        else if word vector (WV) of w exist in GLV
             EM[w] = WV
        end if
     end for
    Return MOV and EM
End
```

3.5 The Selection of User-Based Features

In this dissertation, feature selection methods were applied on the Twitter dataset and the sentimental LIAR dataset to determine the relevant user-based features. Unlike previous studies that used only NLP (text processing) to predict fake news, this dissertation analyzes texts using NLP, on the one hand, and on the other hand, identifies the best user-based features in classifying fake news.

Therefore, selecting these relevant features can improve the accuracy of fake news prediction as well as reduce the dimensionality of the dataset. The relevant user-based features have been selected based on two phases. In *the first phase*, all features were evaluated in three techniques, MDGI, DFI, and correlation methods, independently. *The second phase* is to propose a fuzzy model based mainly on the outputs of the first phase.

3.5.1 Mean Decrease in Gini-index

The behavior of the MDGI method is very similar to the embedded methods for feature selection in that it weighs the features while training the prediction model. The MDGI method exploits the properties of the random forest classifier in constructing trees for feature evaluation. Usually, trees in random forest classifier grow without pruning to the nodes (features). Thus, each feature reduces by a certain percentage the amount of the Gini index when it is selected as a splitting node in the random forest trees.

The proposed method monitors the feature in each tree and calculates its contribution to reducing of Gini-Index criteria. The contribution of the feature to the reduction of the Gini index is calculated by subtracting the amount of the Gini index after selecting that feature as the splitting node from the amount of the Gini index in the dataset before the splitting process. It then calculates the average for the contributions of the feature in all trees as a weight of this feature. For more details see algorithm (3.2).

Algorithm 3.2: MDGI for feature Selection

Input: *Dataset of user-based features*

Output: *Array (MDGI) of two vectors contains features and their weights*

Begin

// Compute Decrease in Gini index to build trees

1. ***for*** *each tree in the forest*
2. ***for*** *f = 1 to n_f* ***do*** *// n_f is a number of features in sample*
3. *Compute Gini for the features: Gini (f) =1-* $\sum_{i=1}^{c} p(i)^2$
   // where c is number of classes and p is probability
4. *Select the best feature (f) as a splitting node*
5. *Calculate the Gini index for its Childs using the above Eq.*
6. *Dec_Gini = Gini(f) – [Gini (left child) + Gini (right child)]*
7. ***end for***
8. ***end for***

// Compute the mean decrease in Gini index for f

9. ***for*** *f = 1 to num_f* ***do*** *// num_f is a number of all features*
10. *f_count = 0*
11. ***for*** *t = 1 to Num_Trees* ***do***
12. *if feature f ∈ tree t*
13. *Dec_Gini[f] = Dec_Gini*
14. *f_count = f_count +1*
15. *end if*
16. ***end for***
17. *Mean_Dec_Gini = Sum (Dec_Gini[f]) / f_count*
18. *MDGI[f] = Mean_Dec_Gini*
19. ***end for***
20. *Return MDGI*

End

3.5.2 Drop-Feature Importance

DFI method evaluates features based on the concept of the wrapper method for feature selection, as it employs a prediction model in weighing features. Using the random forest classifier, this proposed method assigns a weight to the feature according to its contribution to increasing the accuracy. This is achieved as follows:

First, the accuracy of the model is calculated for all features. Then the first feature is removed and the model is re-evaluated. Finally, the difference between accuracy with all features and accuracy after removing the first feature is the weight of that feature. The same goes for all the other features. Algorithm (3.3) shows the steps of the DFI method.

Algorithm 3.3: DFI for feature Selection

Input: *Dataset of user-based features*

Output: *Array (DFI) of two vectors contains features and their weights*

Begin

1. *Set Random Forest Classifier as a Prediction Model*
2. *Training the Model Using Hold-out Method*
3. *Original_Accuracy* = $\frac{TP+TN}{TP+TN+FP+FN}$
4. ***for*** *f = 1 to n* ***do*** *// n is a number of features*
5. *Drop Feature (f)*
6. *Re-trained the Model*
7. *Drop_Accuracy* = $\frac{TP+TN}{TP+TN+FP+FN}$
8. *Feature_Weight = Original_Accuracy - Drop_Accuracy*
9. *DFI[f] = Feature_Weight*
10. *Return the Feature f to the Dataset*
11. ***end for***
12. *Return DFI*

End

3.5.3 Fuzzy Model

The proposed FM identifies relevant user-based features. This model assigns a rank or weight to each feature based on three models: MDGI, DFI, and correlation method. Each of these methods assigns a different weight to the same feature. Thus, the FM unifies the weights by assigning one weight to each feature. Fuzzification and defuzzification were applied as follows:

As shown in Figure (3.4), the triangle membership function shown by equation (2.9) was applied to the feature weights obtained from the above three methods. Thus, each feature now has three weights from the MDGI, DFI, and correlation methods and three other fuzzy values from the triangle membership function.

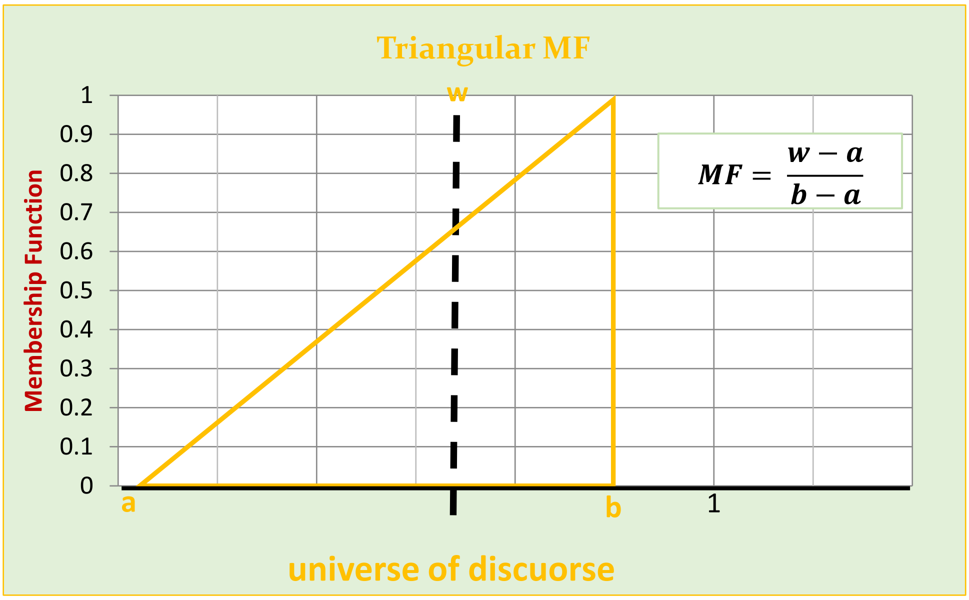


Figure (3.4): Triangle Membership Function

The Center of Gravity method (COG) has been applied in defuzzification to assign one weight for each feature using Equation (2.10). The final weight is obtained through the COG method by relying on three weights and three values of the Membership Functions for each feature. Finally, the relevant features are selected according to a predefined threshold.

Figure (3.5) and Algorithm (3.4) illustrate the most important steps for selecting the relevant user-based features using the FM.

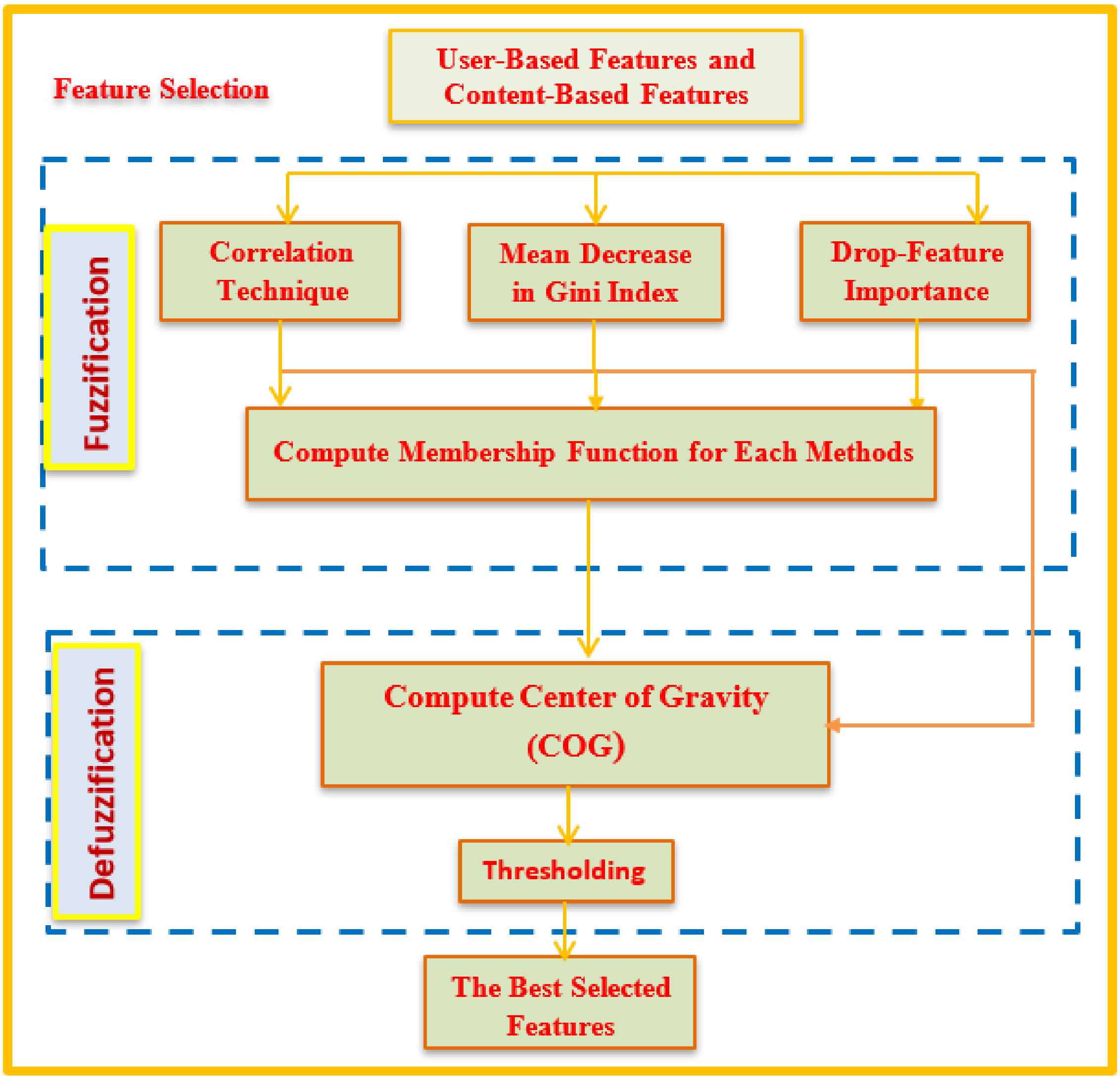


Figure (3.5): FM Architecture

Algorithm 3.4: FM for Feature Selection

Input: *Three vector of features: MDGI, DFI, Corr_F, and Ɵ*

Output: *Array (**Relevant_Features**) contains the Best Selected Features*

Begin

// Computing membership function for the input (the weights of MDGI, DFI, and Corr_F) (fuzzification)

1. ***for*** *f = 1 to num_f* ***do*** *// num_f is a number of features*
2. ***for*** *m =1 to w_f* ***do*** *// w_f = 3 (three weights for each feature)*
3. $MF[m] = \frac{w_m - a}{b - a}$

   // w is the weight of feature. a and b are the triangle coordinates
4. ***end for***

5. ***end for***

// Computing the COG method (defuzzification)

6. ***for*** *f = 1 to num_f* ***do***

7. $COG = \frac{\sum_{i=1}^{n}(MF(w_f) * w_f)}{\sum_{i=1}^{n} MF(w_f)}$ *// n is number of weights for each feature*

8. ***if*** *COG > Ɵ* ***then***

9. *Relevant_Features [f] = COG*

10. ***end if***

11. ***end for***

12. ***Return*** *"Relevant_Features"*

End

3.6 Developed Machine Learning Model

The DMLM focuses on the properties of the random forest classifier and is concerned with modifying its behavior to improve the prediction of fake news. The random forest classifier is an ensemble predictor algorithm based on a set of decision tree models. The disadvantage of this algorithm is that it randomly selects the features in building each tree in the forest. Thus, this algorithm is developed by controlling the selection of user-based features in each tree rather than random selection.

The DMLM is trained on the Twitter dataset and the sentimental LIAR dataset. In the Twitter dataset, each record or object contains the tweet as well as a set of user-based and content-based features (see Appendix A). Similarly, in the sentimental LIAR dataset, each record contains the text or article as well as a set of speaker-based features (see Appendix B). As shown in Figure (3.6), improving the prediction of fake news using these two datasets was conducted in parallel between texts or tweets on the one hand, and user-based features on the other hand.

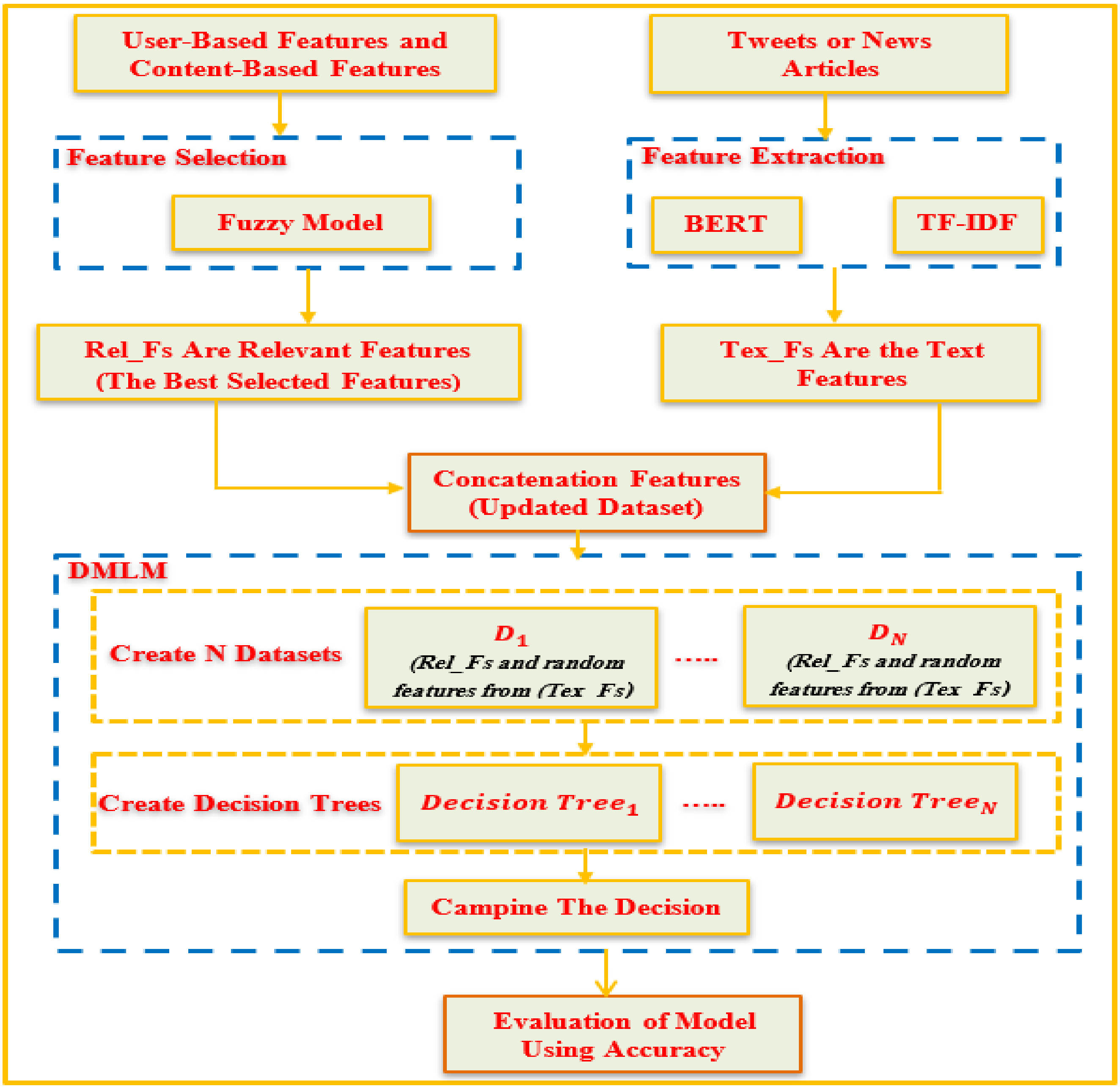


Figure (3.6): DMLM Architecture

In the Twitter dataset, user- and content-based features have been preprocessed by normalizing their values between (0 and 1). Then, these features were evaluated using FM to select the best ones. According to the predefined threshold, the FM reduced these features to (9) out of (48) features.

In contrast, the tweets are represented and converted into features (unique words) by the TF-IDF technique. Since these features are very large, max-features are determined in the TF-IDF model to be 2000. Next, these features (the 9 selected features using FM and the 2000 extracted features by TF-IDF) are concatenated. Based on selected and extracted features, the Twitter dataset became a matrix containing 3000 instances or records and 2009 features.

After that, the random forest classifier is developed by controlling the selection process for features in each sample, which implies that the selection process is not random. The number of trees in the random forest model is set to 100, and 12 features are selected for each tree. Finally, the model chooses the first 5 relevant user- and content-based features (the first 5 features out of the 9 features resulting from the FM) in each tree, whereas randomly chooses the other 7 features resulting from the TF-IDF technique and the last 4 remaining features from FM (i.e., the 4 remaining features from FM are also include in random selection).

In the same way the sentimental LIAR dataset was processed. The only difference is using a BERT pre-trained model for word representation instead of using the TF-IDF method. Each record in the sentimental LIAR dataset contains the article and 23 speaker-based features.

Regarding articles, instead of using TF-IDF with machine learning techniques for word representation, BERT pre-trained model has been used to represent any article (sentence) into vectors. The benefit of using BERT is that this model assigns a vector to each sequence or each sentence. In other words, BERT converts every article or sentence in the dataset into a vector containing embedding values for the entire sentence, not for each word in the sentence. Thus, 12786 vectors are extracted, which is equal to the number of sentences in the Sentimental LIAR dataset. Each of these vectors has a length of 768 embedding values.

The sentimental LIAR dataset contains 23 speaker-based features. The FM was also used to select the best features that improve the prediction accuracy. Therefore, these features were weighed and the five best features are selected based on a predefined threshold. Finally, these five features are merged with the features extracted from BERT pre-trained model. After the merging process, the embedding matrix of this dataset has a size of (12786 *773).

In the same way, the developed random forest is used for FND. The number of features in each tree is configured according to equation (2.27). All forest trees contain the five speaker-based features selected by the FM, while the other required features in each tree are randomly selected from the features extracted by the BERT pre-trained model. The prediction of fake news using DMLM is shown in Algorithm (3.5).

Algorithm 3.5: DMLM for Fake News Prediction

Input: *Pre-processed Dataset (output of algorithm 3.1) and Relevant Features (output of algorithm 3.4)*

Output: *The Prediction Accuracy of Developed Model*

Begin

1.	*Set number of trees N_TREE and number of features N_FEATURE*
2.	*Merge Pre-processed Dataset and Relevant Features*
3.	*Ordering the features, so the Relevant Features are at the beginning and the other extracted features after and saving the indexes of all features in a Matrix (features index). Creating the matrix (selected index) to save the selected features.*
	// Selecting the features in each tree in the forest (DMLM)
4.	*Set Index_RF to zero*
5.	***for*** *i = 1 to N_TREE* ***do***
6.	***for*** *j = 1 to N_FEATURE* ***do***
7.	***if*** *(Index_RF< Θ)* ***Then***
8.	*Selected_Index = Index_RF*

Index_RF= Index_RF+1
Else
Selected_Index = randomly selected index (features index)
end_if
// Shifting the selected features out of matrix
feature index (selected index) = feature index (length – 1)
feature index (length – 1) = selected feature index
length = length – 1
end_for
Building Decision Tree Classifier on (Selected_Index)
end_for
// Training and Testing the Model
Training DMLM
Set ACC_TREES to zero
for** t = 1 to N_TREE **do
Computing the accuracy (ACC) for the tree (t)
ACC_TREES = ACC_TREE + ACC
end_for
Model_Accuracy = ACC_TREES / N_TREE
Return Model_Accuracy
End

3.7 Attention-Based Hybrid Deep Model

The main objective of using the ABHDM is to improve the prediction of fake news in both Arabic and English languages. Each deep neural network has useful properties and may have some drawbacks. A CNN is good at reducing the dimensionality of data and selecting relevant features from the text, but its drawback is that it neglects context. On the other hand, both the Bi-LSTM and the Bi-GRU are distinguished in focusing on the context of the sentence, but they give all features (words) equal importance.

To address these drawbacks of the existing deep neural networks for fake news detection, this dissertation proposes a deep model, ABHDM, for the prediction of both short and long fake news articles. In ABHDM, pre-trained models (FastText and GloVe), CNN, Bi-LSTM, Bi-GRU, and attention mechanisms are used to identify better short-term dependencies, long-term dependencies, and more informative words (relevant features). The proposed deep model is trained on three datasets: the AFND, the AraNews dataset, and the Fake-or-Real dataset. Then the performance of this model was evaluated and its results were promising compared to previous studies conducted on these datasets. Figure (3.7) views the main stages of the proposed model.

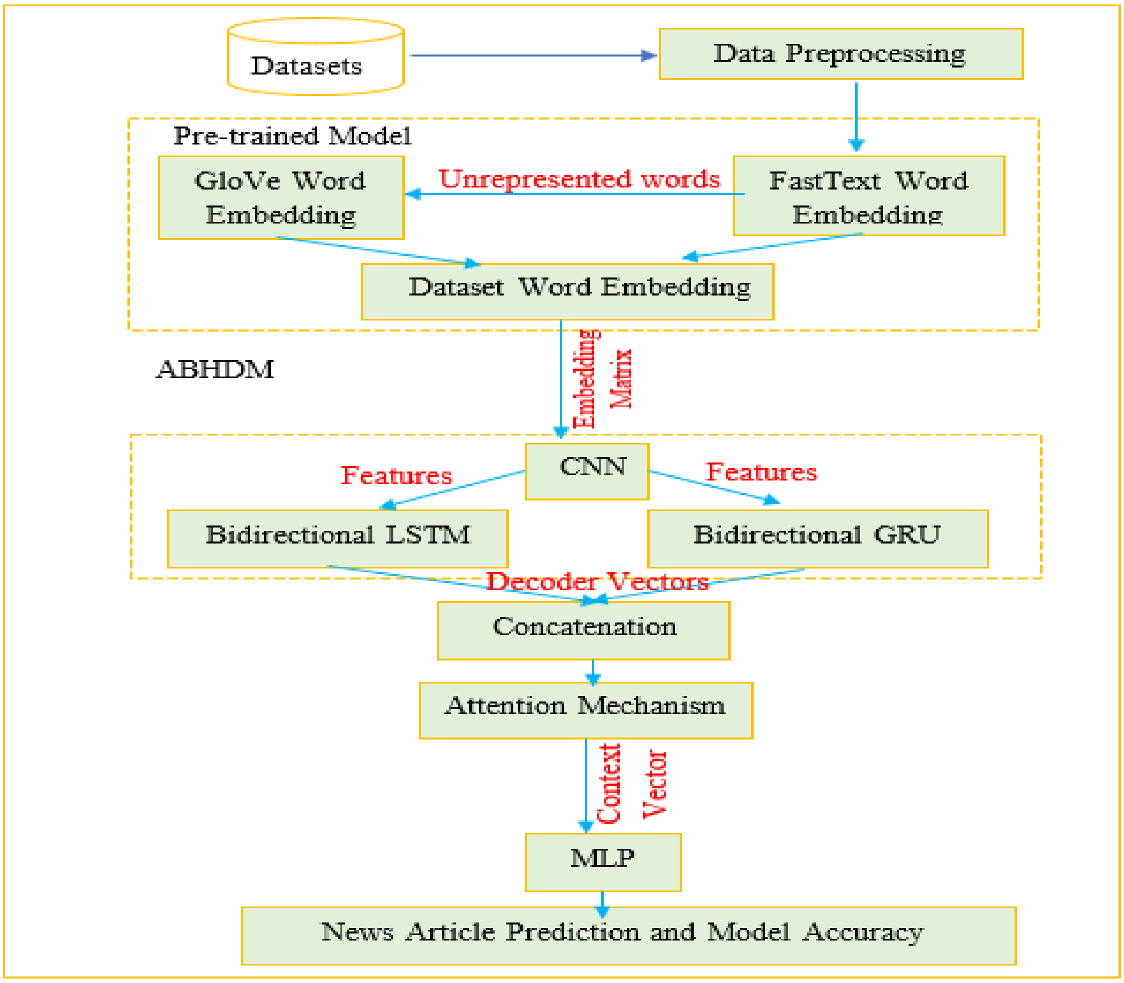


Figure (3.7): ABHDM Stages

As shown in Figure (3.7), five deep neural networks (layer or two from each one) contributed to the prediction of fake news: CNN, Bi-LSTM, Bi-GRU, attention model, and MLP. Initially, the dataset was processed and cleaned as described in the pre-processing section. The words were then represented as described in the feature extraction section. The word representation output represents an embedding matrix that is added as an embedding layer in our proposed model (ABHDM).

On the output of this embedding layer, CNN layers were applied (The CNN is fed with the outputs of the embedding layer). The ABHDM uses two layers of the CNN: convolution layer and max pooling layer. the convolution layer is applied to reduce feature space of dataset, whereas the relevant features are selected by a max pooling layer (see Figure (3.8)).

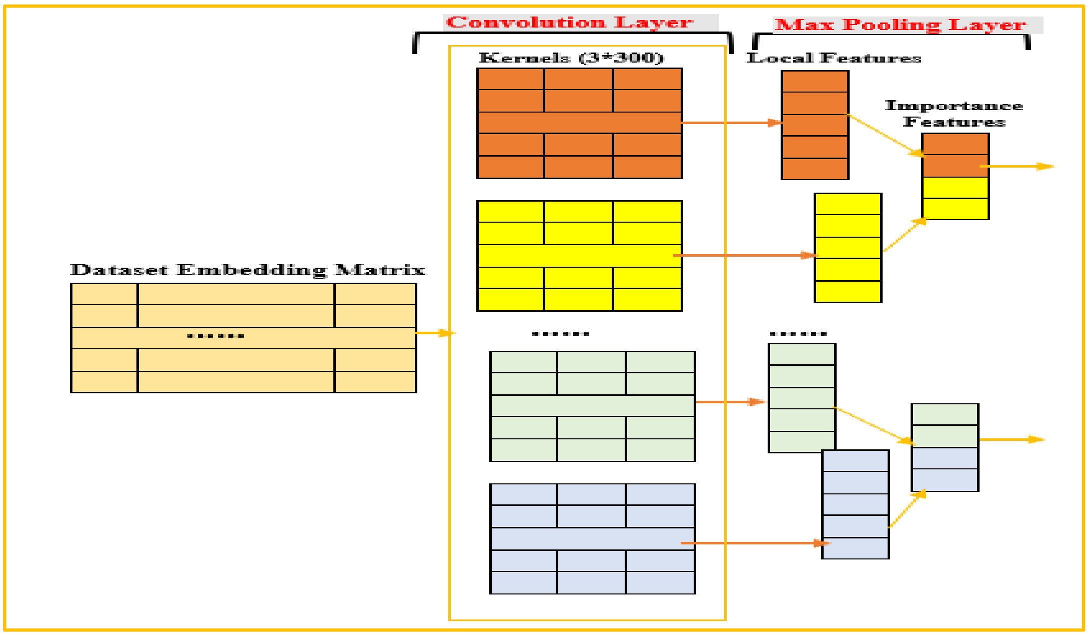


Figure (3.8): The Output CNN Based on of Pre-trained Model

The traditional approach when using CNN in text classification is to add a flattened layer to concatenate the max pooling layer outputs. Then, a fully connected layer or output layer was used to classify the texts. Instead, this dissertation uses both Bi-LSTM and Bi-GRU in parallel (two independent layers) on the output of the max pooling layer. The idea behind this implementation is to make the model capable of remembering both long and short sequences. The following equations achieve this:

$$\overrightarrow{h_{lstm}} = \overrightarrow{LSTM}(max_t), t \in [1-n] \qquad (3.1)$$

$$\overleftarrow{h_{lstm}} = \overleftarrow{LSTM}(max_t), t \in [n-1] \tag{3.2}$$

$$\overrightarrow{h_{gru}} = \overrightarrow{GRU}(max_t), t \in [1-n] \tag{3.3}$$

$$\overleftarrow{h_{gru}} = \overleftarrow{GRU}(max_t), t \in [n-1] \tag{3.4}$$

Where $\overrightarrow{h_{lstm}}$ is a forward context of LSTM, $\overleftarrow{h_{lstm}}$ is backward context of LSTM, $\overrightarrow{h_{gru}}$ is a forward context of GRU, $\overleftarrow{h_{gru}}$ is backward context of GRU, and max_t is the output of max pooling layer.

Then, the backward and forward contexts have been concatenated according to the following equations:

$$h_{LSTM} = [\overleftarrow{h_{lstm}}, \overrightarrow{h_{lstm}}] \tag{3.5}$$

$$h_{GRU} = [\overleftarrow{h_{GRU}}, \overrightarrow{h_{GRU}}] \tag{3.6}$$

The Bi-LSTM produces output independently of the Bi-GRU. These two parallel layers (Bi-LSTM and Bi-GRU) are merged by adding concatenating layer as follows:

$$h_{LSTM\_GRU} = [h_{LSTM}, h_{GRU}] \tag{3.7}$$

The attention mechanism focuses on the more informative words of the sentence (news article) or the best encoder vectors resulting from Bi-LSTM and Bi-GRU layers. Thus, the attention model is applied to the output of the concatenation layer (h_{LSTM_GRU}) to make the proposed model capable of focusing to create effective context vector from entire words in a sentence. This context vector is created by Equations ((2.6), (2.7), and (2.8)).

At this point, we have a vector representing the most important words (relevant features) in the news article. Having obtained this vector, batch normalization is applied to accelerate the network training. Finally, MLP with five layers is used to classify news articles as fake or real. Two of these layers are fully connected layers. The ReLU is employed with these layers as an activation function. To reduce the overfitting, two dropout layers with a certain probability have been added (after each fully connected layer, a dropout layer was added). Finally, the output layer (classification layer) with the sigmoid activation function is applied for the binary classification of news articles. All the above details for ABHDM are shown in a Figure (3.9) and algorithm (3.6).

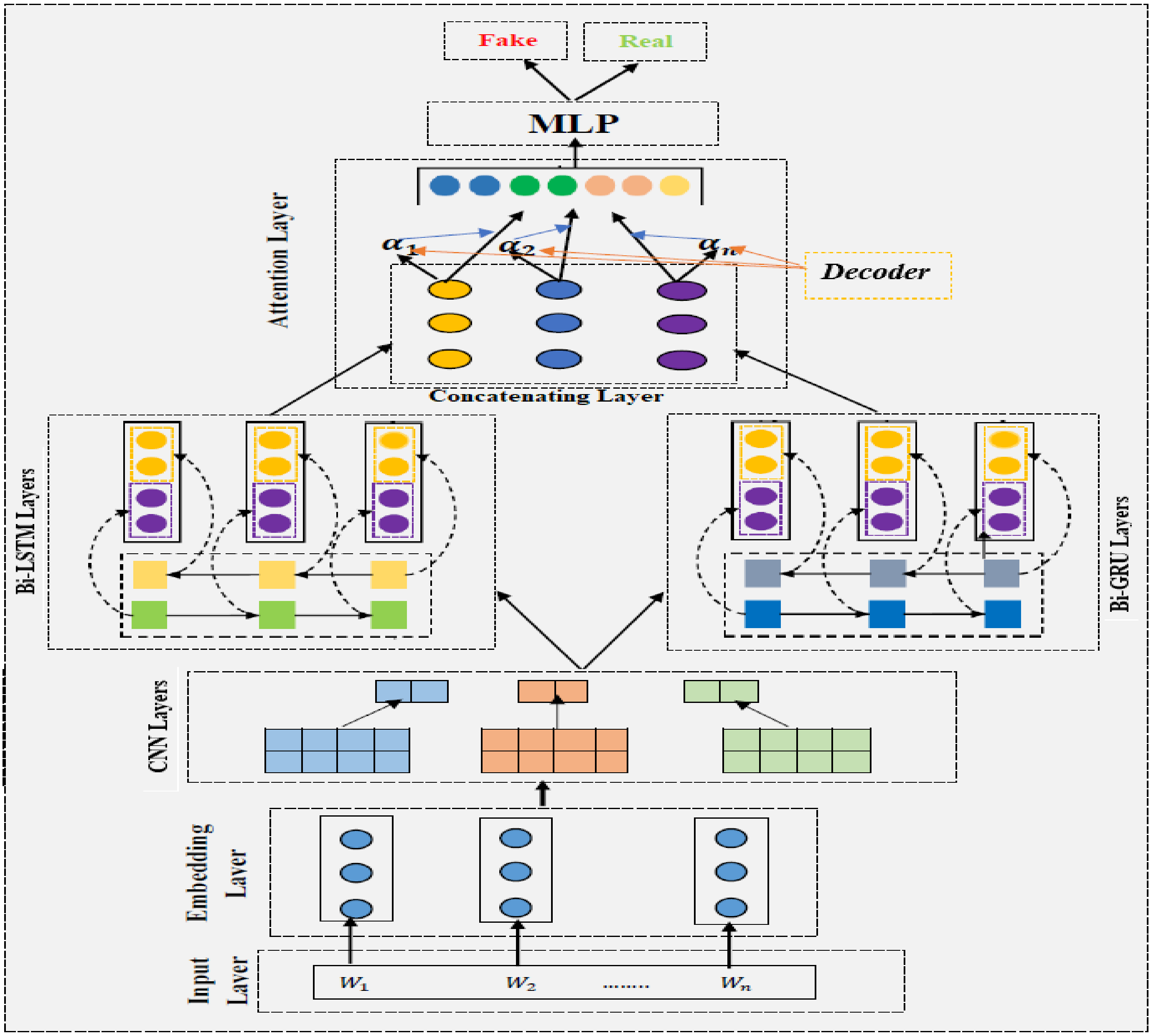


Figure (3.9): ABHDM Architicture

Algorithm 3.6: ABHDM for Fake News Prediction

Input: *Dataset Embedding Matrix* $\boldsymbol{DS} \in \boldsymbol{R}^{n*e}$ *where n is a number of words (top-word) and e is embedding dimension*

Output: *The Prediction Accuracy of ABHDM*

Begin

1. *Initialize the embedding layer with the output of the pre-trained model*
2. *Set max length (max_len), number of filters (n_filters), and filter size (f_size).*

// CNN

3. *num_conv = (max_len) – (f_size) + 1*
4. ***for*** *f = 1 to n_filters* ***do***
5. *i = 0*
6. ***while*** *i < num_conv* ***do***
7. lf_i *= 1D-CNN (DS, f_size)* *// applying Eq. (2.23)*
8. *i = i + 1*
9. ***end_while***
10. rf_f *= max pooling (*lf_i*)* *// applying Eq. (2.24)*
11. ***end_for***

// Bi-LSTM and Bi-GRU

12. *branches = {Bi-GRU, Bi-LSTM}*
13. ***for*** *br ∈ branches* ***do***
14. *applying br on* rf_f *to obtain both forward and backward contexts (*$\overrightarrow{h_{br}}$ *and* $\overleftarrow{h_{br}}$*) by using Eq (3.1), (3.2), (3.3) and (3.4)*
15. *construct* h_{br} *by applying Eq (3.5), (3.6), and (3.7)*
16. ***end_for***

// Attention Model

17. *Applying alignment process on* h_{br} *by using Eq. (2.6)*
18. *Applying softmax to the output of alignment process by Eq. (2.7)*
19. *Finding context vector (cv) by using Eq. (2.8)*

// MLP

20. *Feeding cv into fully connected layer (64 neurons) with ReLU*
21. *Adding dropout Layer (the probability is 0.25)*
22. *Adding fully connected layer (64 neurons) with ReLU*

23. *Adding dropout layer (the probability is 0.25)*
24. *Adding classification layer (one neuron) with sigmoid function*
25. *Updating parameters of the model using the binary cross-entropy loss function with the Adam method and train the network.*
26. *Compute evaluation metric (accuracy) after the training*
27. ***Return Model_Accuracy***

End

3.8 Ensemble Deep Learning Model

This dissertation is concerned with an ensemble approach because the decision-making process is done by taking the opinion of more than one model about the correctness or incorrectness of news articles. The EDLM is an ensemble learning-based model that aims to improve the classification of any news article as fake or real by involving three common text classification techniques in the decision. These three techniques are Bi-LSTM, Bi-GRU, and CNN.

The news articles in the dataset are first pre-processed and tokenized into words. Then, these words (tokens) are converted into embedding matrix of the d dimension using a pre-trained model. Instead of loading random weights, this embedding matrix was load to the embedding layer as weights in the topology of the network and make it non-trainable layer. After that, the output of the embedding layer is fed to both Bi-LSTM, Bi-GRU, and CNN where each of these three deep neural networks gives its decision with the probability that the news article is fake or real.

The necessary steps performed in the construction of the topology for both Bi-LSTM and Bi-GRU are similar. The Bi-LSTM layer is applied to the output of the embedding layer to address the sequences of arbitrary length and extract long and short dependencies in both backward and forward directions. The ReLu activation function was adapted with the Bi-LSTM layer. The output of this layer is fed to the attention layer. The idea behind using the attention layer is to find the best vector in the news article that represents the most informative words in this article. The important vector extracted through the attention layer is passed to the last layer in this network, which is the output layer. Through the sigmoid activation function, this layer produces an estimated value ($\hat{y}$) that represents the probability that the news article is fake or real. The Bi-GRU applies exactly the same procedure. The only difference is that text processing and extracting long and short dependencies are done by the Bi-GRU instead of by the Bi-LSTM. The Bi-GRU produce ($\hat{\hat{\hat{y}}}$) which represents the probability that the news article is fake or real according to the opinion of this deep network.

Two layers of CNN (convolution layer and max pooling layer) with MLP is used to produce ($\hat{\hat{y}}$). The output of the embedding layer is fed into the convolution layer to reduce the feature space. ReLu activation function and many filters with fixed sizes are used in this layer. The max pooling layer is applied on the top of the convolution layer to select the relevant features. To obtain the decision of the CNN, three layers have been used: a fully connected layer containing many nodes with a ReLu activation function, a dropout layer to reduce overfitting, and an output layer with a sigmoid activation function to calculate the estimated value that determines whether the news article is fake or real.

So far, we have three decisions ($\hat{y}, \hat{\hat{y}}, and\ \hat{\hat{\hat{y}}}$) for each news article. Therefore, the concatenation layer was added to concatenate these values and then fed to an output layer with a sigmoid activation function to classify any news article. The descriptive algorithm (3.7) and Figure (3.10) and (3.11) shows the most important steps conducted to classify news articles by using the EDLM.

Algorithm (3.7): EDLM for Fake News Prediction

Input: *Dataset Embedding Matrix* ***DS***

Output: *The Prediction Accuracy*

Begin

1. *Set number of filters, filter size, and number of units for Bi-LSTM, Bi-GRU, and fully connected layer (Dense Layer)*

2. *Loading the embedding matrix (DS) to the embedding layer and making it non-trainable*

3. *On the output of embedding layer do*

// Bi-LSTM

3.1 *Build Bi-LSTM Network through*

3.1.1 add Bi-LSTM layer

3.1.2 add attention layer

3.1.3 add output layer to find $\hat{\boldsymbol{y}}$

3.2 *Build Bi-GRU Network through*

3.2.1 add Bi-GRU layer

3.2.2 add attention layer

3.2.3 add output layer to find $\hat{\hat{\hat{\boldsymbol{y}}}}$

3.3 *Build using CNN Network through*

3.3.1 add convolution layer

3.3.2 add max pooling layer

3.3.3 Add flatten layer to sort the output of max pooling

// Adding Three Layers for Prediction

3.3.4 The output of flatten layer is fed to dense layer

3.3.5 Adding dropout layer (the probability is 0.25)

3.3.6 Adding output layer to find $\hat{\hat{\boldsymbol{y}}}$

// Concatenate the three outputs

4	*Adding concatenation layer to concatenate (*$\hat{y}$ *,* $\hat{\hat{y}}$*, and* $\hat{\hat{\hat{y}}}$ *)*
	// Add Classification Layer
5	*Feeding the output of concatenation layer into output layer and fitting the model to classify the news article*
	// Train the Model and Update the Parameters
6	*Updating parameters of the model using the binary cross-entropy loss function with the Adam method and train the model.*
7	*Compute evaluation metric (accuracy) after the training*
8	***Return Model_Accuracy***

End

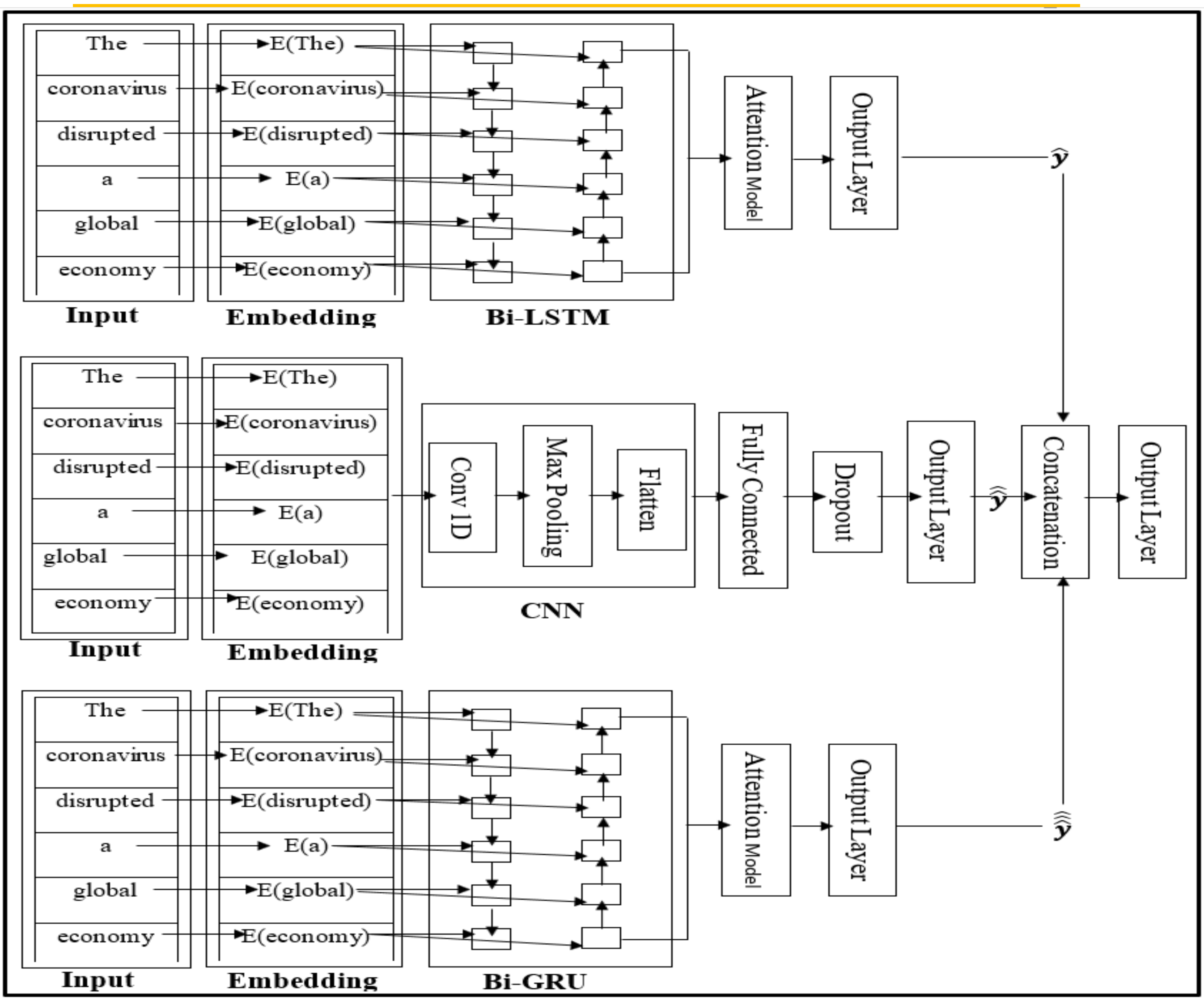


Figure (3.10): The EDLM Layers Based on Single News Article

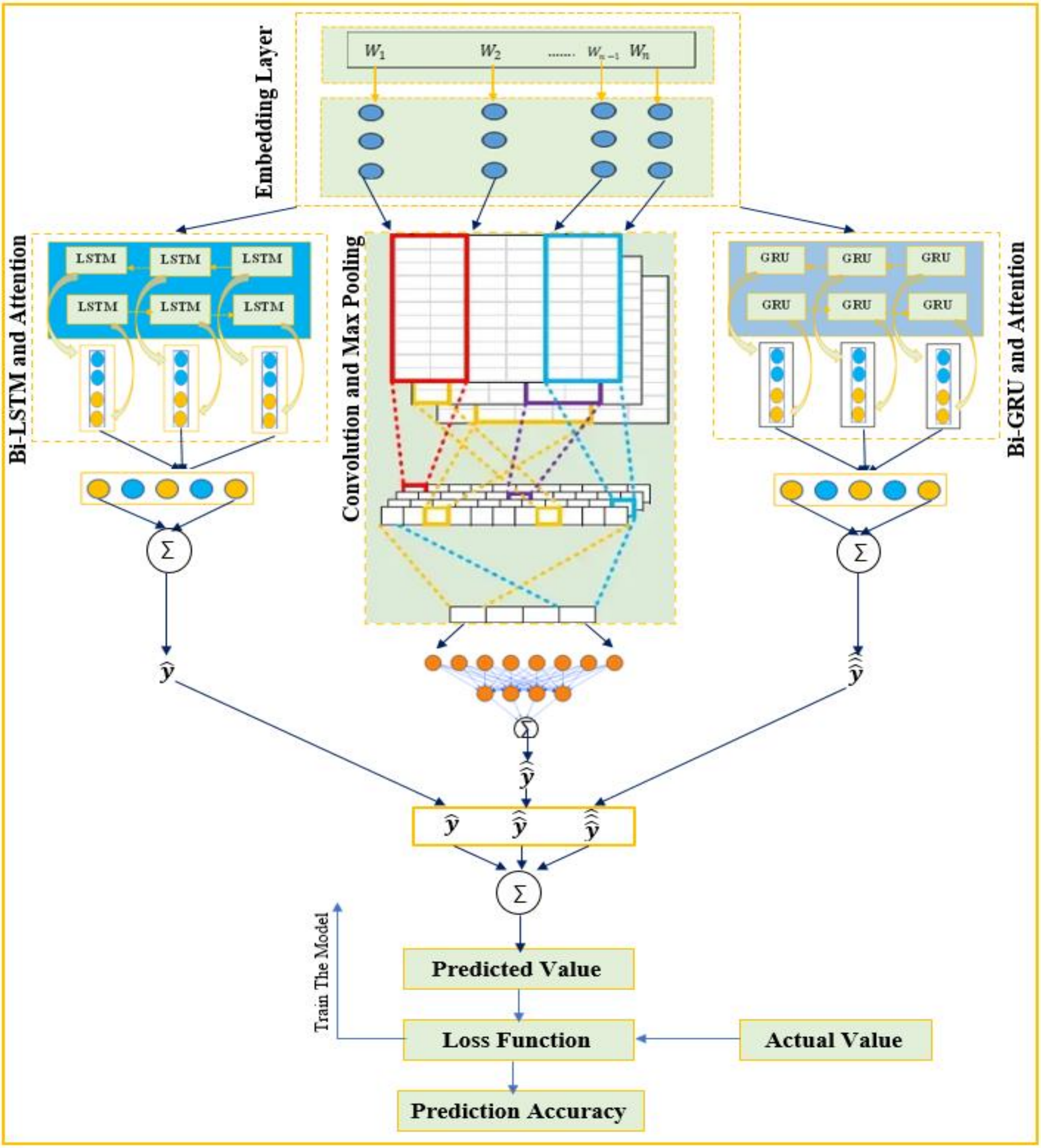


Figure (3.11): EDLM Architecture

3.9 Evaluation of the Proposed Models

The evaluation stage is the most important part of the proposed system to determine the quality and efficiency of our proposed models in detecting Arabic and English fake news. Because our proposed models aim at binary classification, the evaluation metrics used for this task are Accuracy, Precision, Recall, and F-measure. The concepts k-fold cross validation and holdout train-test splitting are employed to calculate these evaluation metrics. In the proposed deep models, the datasets were partitioned as 80:20 to train and test the model. In contrast, the 10-fold cross validation method is applied to train and test the DMLM.

Chapter Four

Results and Discussion

Results and Discussion

4.1 Overview

This Chapter views and discusses the results of the proposed system stages presented in Chapter three. Experiments have been conducted on five datasets available to the research community: Twitter Dataset, AraNews Dataset, AFND, Sentimental LIAR, and Fake-or-Real Dataset. Then, the results are arranged based on their appearance in the previous Chapter. However, this Chapter begins with the software and hardware requirements in implementing the proposed models.

4.2 System Requirements

The specifications (software and hardware) that have been utilized to implement the proposed system are as follows:

1- Hardware:

- PC: (HP).
- Processor: Intel(R) Core i7 CPU, speed 2.70GHz.
- RAM: the memory capacity of 8.0 GB
- Hard: storage 250 GP SSD.

2-Software:

- Windows 10 64-bit.
- Python for executing deep learning techniques.
- JAVA for executing machine learning techniques.
- Anaconda Navigator and NetBeans.
- Library (Tensor flow, Keras, NLTK, and WEKA).

4.3 Description of Datasets

This dissertation uses five datasets available to the research community. Three of these datasets are in Arabic: Twitter, AraNews, and AFND. The remaining two are English language datasets: The sentimental LAIR dataset and the Fake-or-Real dataset.

The Twitter dataset contains exactly 3,000 tweets categorized as reliable (not fake) or unreliable (fake). The number of fake tweets is 1400, whereas the number of real tweets is 1600. In addition to a tweet, each record or instance in this dataset contains a set of user-based features and content-based features (see Appendix A). This dissertation aims to select relevant features as well as analyze the tweet itself in order to improve prediction accuracy. Figure (4.1) shows the most frequent words (word cloud) for a sample of the Twitter dataset (word cloud).

Figure (4.1): The Most Frequent Words in Twitter Dataset

The AraNews dataset is a large dataset collected from many countries and includes comprehensive topics on politics, economics, health, sports, and others. Because this dataset contains many noise and redundant articles, preprocessing was performed. After conducting a preprocessing that includes removing noise, duplicate articles, meaningless articles, and short news articles that do not give full meaning, the number of news articles has reached approximately (16,600). The number of fake news articles is 8406, while the number of non-fake news articles is 8194. Figure (4.2) views the most frequent words for a sample of the AraNews dataset.

Figure (4.2): The Most Frequent Words in AraNews Dataset

AFND consists of 134 different files in JSON format. These JSON files are gathered from 134 public Arabic news websites. The number of news articles in these files is exactly 606912, classified into three categories: credible, undecided, or not credible. Each object in the JSON files stores the article's text, title, and publication date. The text and title are considered in each object, while the date is omitted. According to the previous and comparative study in this dissertation, news articles classified as credible and not credible were relied upon, while the third category (undecided) was neglected. The number of credible news articles is 167233, while the number of not credible news articles is 207310. Finally, all JSON files were combined into a single CSV file, the details of which are shown in Figures (4.3).

Figure (4.3): The Most Frequent Words in AFND

Sentimental LIAR is a modified version of the LIAR dataset containing multi-class labeling. In addition to texts, this dataset contains features that are specific to the speaker or writer. The dataset is converted to a binary class by changing false, barely-true, pants-fire, and half-true labels to *Fake* and the remaining labels to *Not-Fake*. The rationale behind this procedure is that the performance of our proposed models is compared with previous studies, which also transformed the classification into binary in this way. The number of fake examples is 7278, while the number of non-fake examples is 5508. For more details, see Figures (4.4).

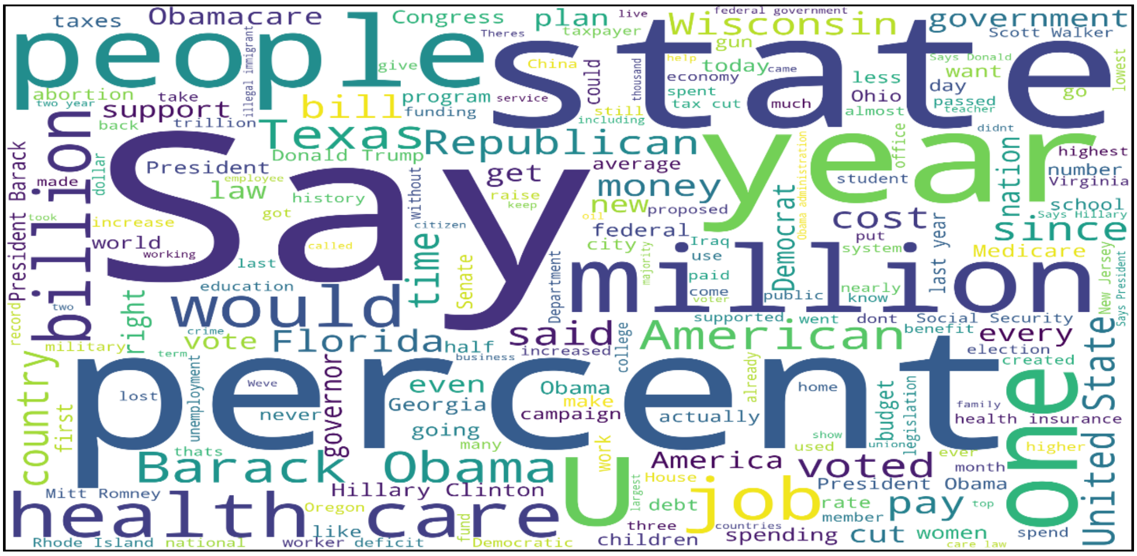


Figure (4.4): The Most Frequent Words in Sentimental LIAR Dataset

The Fake-or-Real Dataset contains two CSV files, the first contains the fake news, and the second contains the real news. Each news article consists of a text, title, article type, article date, and class label. The text and title were considered in the dissertation. The number of fake news articles is 23481, while the number of non-fake news articles is 21417. Figure (4.5) shows the word cloud or the most frequent words in this dataset.

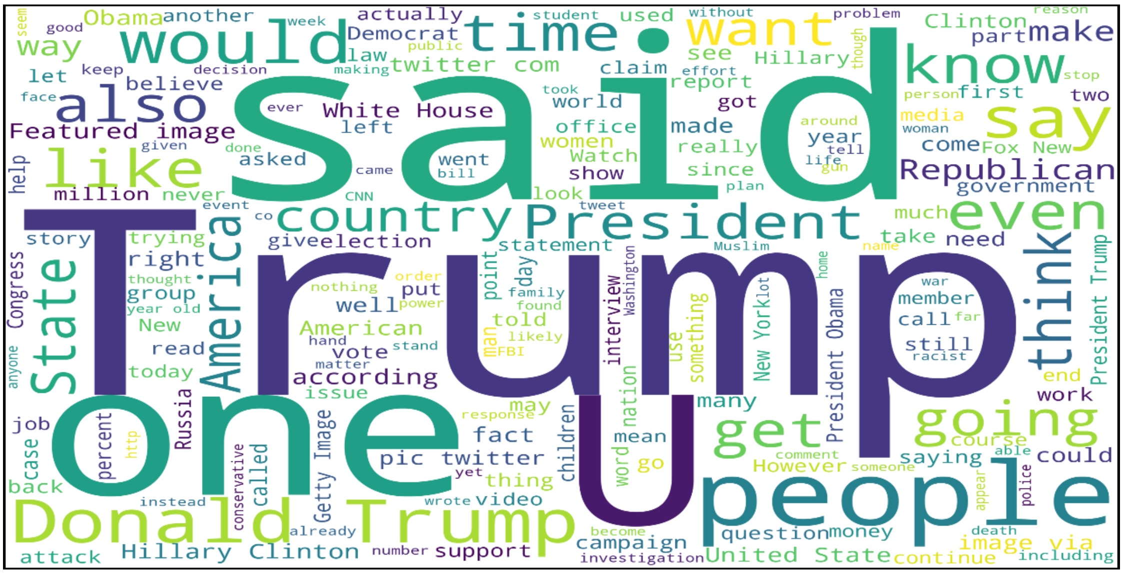


Figure (4.5): The Most Frequent Words in Fake-or-Real Dataset

4.4 Results of Data Preprocessing

The main objective of this stage is to prepare the datasets in an acceptable format for machine learning algorithms. The Preprocessing steps mentioned in Section (3.3) are illustrated in Figure (4.6). This Figure shows the general framework for the preprocessing stage, where additional steps depending on the dataset's type and nature.

For example, the deletion of short tweets is not performed with the Twitter dataset and the sentimental LIAR dataset because most tweets or sentences of these datasets are short. In contrast, two other preprocessing steps were taken with these two datasets: the deletion of stop words and stemming. Likewise, with the AraNews dataset, an additional preprocessing step was taken: removing incomplete news articles that do not meet the news event criteria and removing repeated and short news articles (see Table (3.3)). After cleaning the datasets from the unwanted information, the next step is tokenization. In this step, each news article or tweet was divided into tokens (words) based on the spaces between the words.

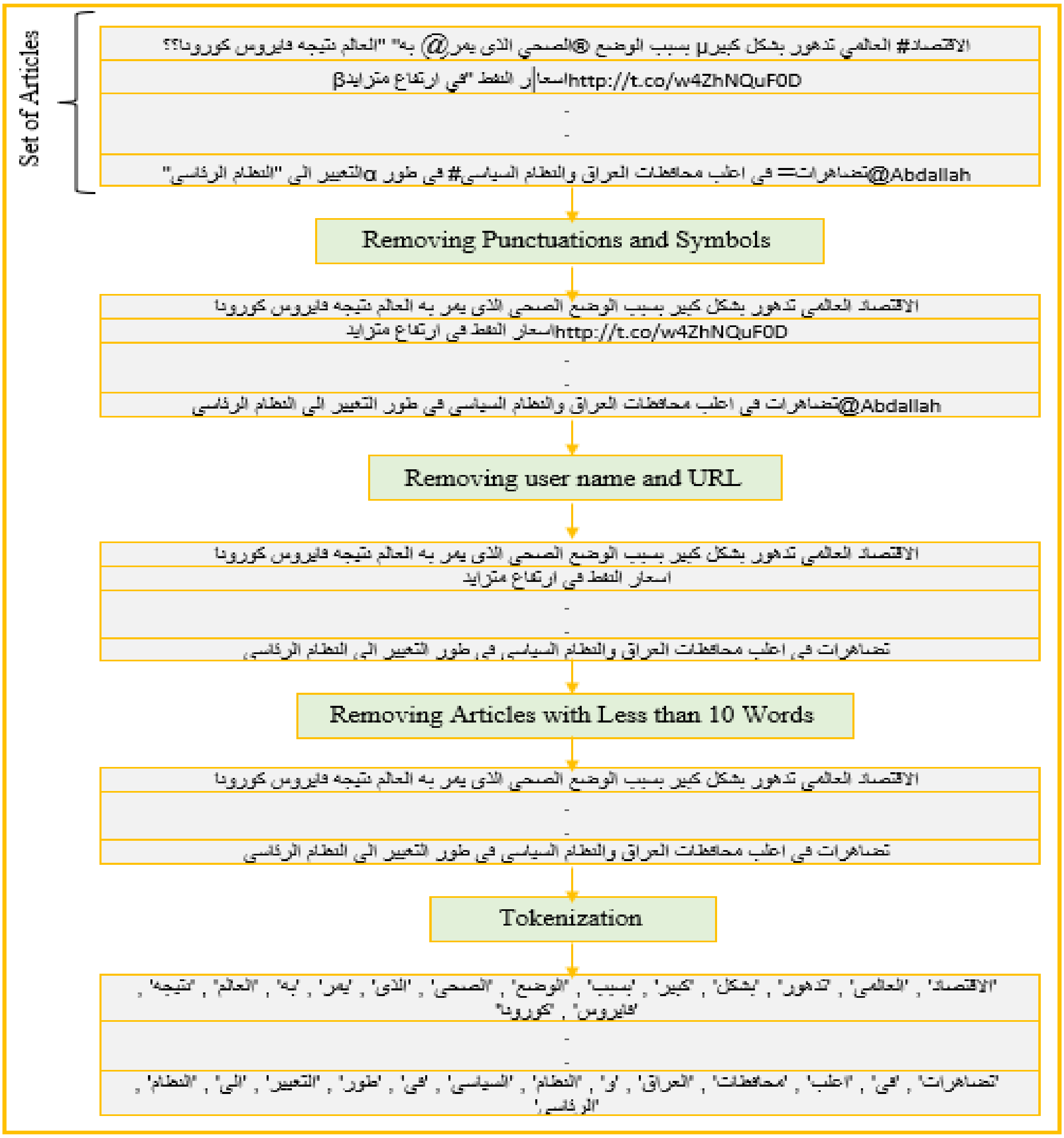


Figure (4.6): Data Preprocessing Steps

4.5 Results of Feature Extraction

In fact, feature extraction is a part of the data preprocessing when dealing with text. Feature extraction techniques aim to represent words and convert them into features or vectors to be fed into machine learning or deep learning techniques. Thus, this stage aims to convert the words or tokens obtained from the previous section (the preprocessing section) into features. Before representing words and converting them into vectors, it is necessary to initialize three basic parameters that effectively improve prediction accuracy: max_words, max_len, and emb_dim. These parameters are initialized based on the nature of the dataset being used.

In practice, focusing on the most important words in the dataset and neglecting the less important words is important. Therefore, the *max_words* parameter determines the most important unique words in the dataset or the top most common words. Table (4.1) presents the specific value of this parameter for the datasets used in this dissertation.

Table (4.1): The max_words parameter initialization

Dataset / Parameter	Twitter dataset	AraNews dataset	AFND Dataset	Sentimental LIAR dataset	Fake-or-Real Dataset
max_words	2000	7000	10000	4000	6000

When the numeric sequence representations of the text are encoded, the news article's lengths will not be uniform. Thus, the *max_len* parameter is needed to specify a fixed length for all news articles. Therefore, any news article whose word count is less than this parameter is padded with zeros either at the beginning (pre-padding) or at the end (post-padding) of the news article. Likewise, in any news article whose word count exceeds the value assigned to the *max_len* parameter, the words are truncated, either from the beginning of the news article or from the end of the article. The *max_len* parameter is usually initialized through three mechanisms: (1) the parameter is equal to the length of the longest news article in the dataset; (2) it is equal to the average length of all news articles; or (3) it is initialized through trial and error. Table (4.2) shows the value of the *max_len* parameter with the datasets. For the Twitter dataset, the TF-IDF was used, which does not need this parameter.

Table (4.2): The max_len Parameter Initialization

Dataset / Parameter	AraNews dataset	AFND Dataset	Sentimental LIAR dataset	Fake-or-Real Dataset
max_len	80	128	25	50
Padding and truncating	Post	Post	Post	Post

Each word in a news article is represented by a vector containing embedding values. The number of values representing each word (size of this vector) is determined by initializing the *emb_dim* parameter. Therefore, the size of the feature or vector depends on the feature extraction technique used with the datasets.

TF-IDF has been applied to the Twitter dataset for feature extraction. This technique assigns a vector (feature) to the tweet as a whole, not to each word. Thus, the size of the extracted feature is equal to the number of unique words specified by the max_words parameter (this means the size of the extracted feature is equal to 2000). Table (4.3) shows a sample of the features extracted by TF-IDF. The defect in traditional methods, including the TF-IDF, is that it does not take into account the semantic meaning of the word in the process of extracting features. Therefore, most of the vectors resulting from these methods have zeros values. This problem was addressed through the use of embedding methods or pre-trained models.

Table 4.3: Features Extraction Using TF-IDF

Tweets \ Features	*Feature* $_1$	*Feature* $_2$	*Feature* $_3$	……	*Feature* $_{2000}$
	آثار	النظام	فايروس	……	سوريا
Tweet $_1$	0.294846	0	0	……	0
Tweet $_2$	0	0	0.369472	……	0
Tweet $_3$	0	0.530210	0.192375	……	0
Tweet $_4$	0	0	0	……	0.5726575
Tweet $_5$	0	0.634421	0	……	0

The features of the Sentimental LIAR dataset were extracted using BERT pre-trained model. Similar to TF_IDF, this model allocates a vector at the sentence level, not the word level. BERT base version is used in the dissertation; thus, a vector of size 768 is assigned to each sentence in this dataset. Table (4.4) shows a sample of the features extracted by the BERT pre-trained model.

Table 4.4: Features Extraction Using BERT

Articles \ Features	*Feature* $_1$	*Feature* $_2$	*Feature* $_3$	……	*Feature* $_{768}$
Sentence $_1$	0.004107	-0.06822	-0.09841	……	-0.01641
Sentence $_2$	-0.13417	0.056375	-0.45205	……	-0.20983
Sentence $_3$	-0.02755	-0.08062	-0.34574	……	-0.27301
Sentence $_4$	-0.00713	0.013269	-0.12596	……	-0.00537
Sentence $_5$	0.004107	-0.06822	-0.09841	……	-0.01641

The Keras embedding layer and pre-trained models were adapted to extract features in the AraNews dataset, the AFND, and the Fake-or-Real Dataset. The Keras embedding layer was used to extract features for these three datasets by creating a random embedding matrix for this layer and learning it during training (dynamic training). The emb_dim parameter for this layer is set to 150.

On the other hand, pre-trained models are the research community's modern trend through their feature extraction use. Among the most important of these models that deal with the Arabic and English languages are FastText and GloVe, respectively. Unlike BERT, which assigns a vector to the whole sentence, these models allocate a vector to each word by pre-training on millions of words. The FastText pre-trained model was used to extract the features of the AraNews dataset and the AFND, while the GloVe pre-trained model was applied to the Fake-or-Real Dataset. Each word is represented by these two models with a vector containing 300 embedding values (this means that the *emb_dim* parameter is equal to 300). This dissertation uses the embedding matrix generated by these two models and uploads it to the embedding layer. Then, this layer was added directly after the input layer in the proposed deep model architecture, which will be clarified later. Because the outputs of this layer are the outputs of the pre-trained models, we have made this layer untrainable (static training). Table (4.5) provides more details about this section.

Table (4.5): The Results of Feature Extraction

Dataset / Parameter	Twitter dataset	AraNews dataset	AFND	Sentimental LIAR dataset	Fake-or-Real Dataset
Feature Extraction Technique	TF-IDF	FastText and Keras embedding layer	FastText and Keras embedding layer	BERT	GloVe and Keras embedding layer
emb_dim Parameter	------	300: FastText 150: embedding	300: FastText 150: embedding	------	300: GloVe 150: embedding
The Size of Embedding Vectors (The Output)	[3000*2000]	[7000*300] [7000*150]	[10000*300] [10000*150]	[12786*768]	[6000*300] [6000*150]

4.6 Results of Feature Selection

Most of the previous studies interested in FND depend mainly on NLP to determine the credibility of the news and neglect some important features of the user. In contrast to these studies, user-based features are considered with the texts for FND in this dissertation. In addition to the sentence or the tweet, the Twitter and Sentimental LIAR datasets contain several features specific to the user or speaker. However, not all of these features are suitable for the prediction task, as some of them do not improve the model's performance and may reduce the accuracy of the prediction. For this reason, this dissertation focuses on identifying relevant features only and excluding irrelevant features.

4.6.1 The Best Selected Features of the Twitter Dataset

Appendix A shows that the Twitter dataset contains exactly 48 features. To identify the relevant features, a reliable fuzzy model based on three different approaches for feature selection has been proposed: filter, embedded, and wrapper approaches. In the filter approach, the correlation technique has been applied to evaluate and weigh user-based features according to their importance and relevance to the target class. Table (4.6) explain the user-based features and their weights according to the correlation method.

Table 4.6: Rank the User-Based Features According to Correlation Technique

Features	Weight	Features	Weight
count_of_followers	0.311	registration_age	0.078
tweet_is_a_reply	0.300	has_negative_words	0.071
count_of_special_symbols	0.269	has_positive_words	0.071
count_of_mentions	0.260	length_of_screen_name	0.064
status_count	0.250	count_of_urls	0.064
length_of_tweet_in_words	0.228	favorites_count	0.056
has_hashtag	0.227	average_urls_mentions	0.055
has_url	0.222	day_of_week	0.049
count_of_unique_words	0.208	count_of_ellipses	0.037
count_of_unique_chars	0.191	negative_sentiment_score	0.030
count_of_hashtags	0.180	focus_of_user_on_topic	0.026
retweet_fraction	0.175	has_user_mention	0.026
listed_count	0.148	tweet_time_spacing	0.025
Fefo	0.148	has_default_image	0.024
has_exclamation_mark	0.139	has_question_mark	0.016
average_number_of_hashtags	0.130	count_of_question_marks	0.016
length_of_description	0.128	retweeted	0.013
Fofe	0.124	positive_sentiment_score	0.012
count_of_exclamation_marks	0.113	count_of_friends	0.012
is_verified	0.110	count_of_retweets	0.012

has_description	0.106	average_favorite_count	0.010
length_of_tweet_in_chars	0.093	used_url_shortner	0
average_retweet_count	0.089	tweet_is_a_retweet	0
average_tweet_length	0.083		

In a manner similar to the embedded approach, this dissertation proposes an MDGI that evaluates user-based features according to each feature's contribution to reducing the Gini index in dataset. These features are implicitly evaluated and weighted with the random forest classifier. Each feature reduces the impurity (Gini Index) in the sample chosen from the original dataset to construct the forest decision tree by a certain percentage when it is selected as a splitting node. Thus, the feature that contributes the most to reducing the Gini index by taking the average contributions in all decision trees has a higher rank (see Table (4.7)).

Table 4.7: Rank the User-Based Features According to MDGI

Features	Weight	Features	Weight
count_of_friends	0.460	fefo	0.260
count_of_followers	0.440	length_of_tweet_in_chars	0.260
has_hashtag	0.430	length_of_screen_name	0.250
has_default_image	0.420	positive_sentiment_score	0.250
length_of_tweet_in_words	0.410	fofe	0.250
registration_age	0.400	count_of_mentions	0.240
count_of_unique_words	0.390	focus_of_user_on_topic	0.240
count_of_retweets	0.360	favorites_count	0.240
Retweeted	0.350	has_exclamation_mark	0.230
count_of_special_symbols	0.350	length_of_description	0.230
has_user_mention	0.340	listed_count	0.230
count_of_hashtags	0.320	status_count	0.220
has_url	0.320	tweet_time_spacing	0.220
count_of_unique_chars	0.300	retweet_fraction	0.220
average_favorite_count	0.300	has_question_mark	0.220
average_number_of_hashtags	0.300	has_description	0.220
count_of_urls	0.300	tweet_is_a_reply	0.210
has_negative_words	0.290	count_of_question_marks	0.200
has_positive_words	0.290	count_of_ellipses	0.180
average_retweet_count	0.290	negative_sentiment_score	0.160
average_tweet_length	0.290	is_verified	0.150
average_urls_mentions_ratio_in_tweets	0.280	used_url_shortner	0.000
count_of_exclamation_marks	0.280	tweet_is_a_retweet	0.000
day_of_week	0.270		

The DFI is a method similar to or close to the wrapper approach that weighs user-based features based on the contribution of each one in increasing accuracy. Through this method, a weight is assigned to each feature as follows: (1) tuning the random forest classifier to calculate prediction accuracy using all user-based features; (2) dropping the first feature, and then the prediction accuracy is calculated after dropping this feature; (3) calculating the difference between the two accuracies, as the resulting value determines the contribution of the feature in increasing the accuracy, noting that the greater this value, the more important the feature; (4) finally, the feature is reverted to the original features, and then these steps are applied to all remaining features. Table (4.8) presents the importance degree of user- and contianed-based features.

Table 4.8: Rank the User-Based Features According to DFI

Features	Weight	Features	Weight
tweet_is_a_reply	0.169	average_urls_mentions	0.027
count_of_retweets	0.113	has_exclamation_mark	0.026
has_user_mention	0.108	average_number_of_hashtags	0.024
count_of_mentions	0.101	length_of_screen_name	0.023
count_of_followers	0.065	count_of_unique_chars	0.020
count_of_hashtags	0.056	average_tweet_length	0.019
registration_age	0.056	average_favorite_count	0.017
count_of_friends	0.048	length_of_description	0.015
count_of_unique_words	0.046	focus_of_user_on_topic	0.012
average_retweet_count	0.046	length_of_tweet_in_chars	0.011
tweet_time_spacing	0.046	has_description	0.011
Fefo	0.043	has_negative_words	0.010
count_of_ellipses	0.042	has_positive_words	0.010
retweet_fraction	0.039	has_default_image	0.010
favorites_count	0.039	positive_sentiment_score	0.000
has_hashtag	0.038	negative_sentiment_score	0.000
count_of_exclamation_marks	0.038	has_question_mark	0.000
length_of_tweet_in_words	0.037	tweet_is_a_retweet	0.000
count_of_urls	0.037	Retweeted	0.000
has_url	0.036	used_url_shortner	0.000
listed_count	0.035	day_of_week	0.000
Fofe	0.034	count_of_question_marks	0.000
status_count	0.033	count_of_special_symbols	0.000
is_verified	0.032		

So far, each user-based feature has three different weights from the three methods above. These vectors containing the weights of the features were fed to the FM in order to have one weight for each feature. The proposed fuzzy model includes two main steps: In the fuzzification step, the triangular membership function was used to calculate the fuzzy value (the belonging degree). Thus, each feature has three weights and three membership degrees. One weight was calculated for each feature using the COG in the defuzzification step.

After the importance of the features relative to the target class was determined according to the FM and the user-based features were weighted and arranged from best to least, any feature weighing less than or equal to (0.3) was deleted. This procedure reduced the features to only 9 out of the 48. Table (4.9) presents the feature weights according to the FM.

Table 4.9: The Relevant User-Based Features According to FM

Features	Description	weight
count_of_friends	The number of friends	0.412
has_default_image	Does the account contain a personal	0.403
count_of_followers	The number of followers	0.361
count_of_retweets	The number of retweets	0.349
has_hashtag	Does the tweet contain a hashtag?	0.342
length_of_tweet_in_words	The number of words in tweet	0.328
has_user_mention	The account has a name mentioned	0.318
registration_age	The age of user	0.317
count_of_unique_words	The number of unique word in tweet	0.307

Table 4.9 shows the most important features affecting the target class. Therefore, this dissertation recommends that decision-makers consider these features when verifying any tweet's reliability.

4.6.2 The Best Selected Features in Sentimental LAIR Dataset

The Sentimental LIAR dataset also contains several speaker-based features (see Appendix B). To determine the relevant features, the proposed fuzzy model applied to the Twitter dataset was also applied to this dataset. Before the proposed feature selection models evaluated the speaker-based features, some non-predictive features, such as ID, were manually omitted. Tables (4.10), (4.11), and (4.12) view the rank (weights) of the remaining speaker-based features according to correlation technique, MDGI, and DFI respectively.

Table 4.10: Features Rank of Sentimental LIAR According to Correlation Technique

Features	Weight
Sentiment	0.157
party_affiliation	0.108
false_counts	0.101
Subject	0.066
Anger	0.064
speaker_job	0.050
sentiment_score	0.046
Sad	0.043
pants_on_fire_counts	0.041
state_info	0.038
Disgust	0.035
mostly_true_counts	0.024
Joy	0.024
barely_true_counts	0.022
half_true_counts	0.022
Fear	0.018
sentiment_magnitude	0.005

Table 4.11: Features Rank of Sentimental LIAR According to MDGI

Features	Weight
speaker_job	0.400
Sentiment	0.300
Subject	0.300
Sad	0.290
sentiment_magnitude	0.280
Anger	0.280
Fear	0.270
Joy	0.260
Disgust	0.260

false_counts	0.250
state_info	0.250
sentiment_score	0.250
barely_true_counts	0.230
half_true_counts	0.220
mostly_true_counts	0.220
party_affiliation	0.210
pants_on_fire_counts	0.200

Table 4.12: Features Rank of Sentimental LIAR According to DFI

Features	Weight
Subject	0.100
pants_on_fire_counts	0.090
Fear	0.035
mostly_true_counts	0.021
barely_true_counts	0.019
speaker_job	0.016
Sad	0.011
false_counts	0.007
state_info	0.006
Sentiment	0.002
Disgust	0.000
half_true_counts	0.000
Joy	0.000
Anger	0.000
sentiment_magnitude	0.000
sentiment_score	0.000
party_affiliation	0.000

Finally, the relevant features were determined according to the proposed FM where any feature whose weight is greater than or equal to (0.2) is selected, and all other features are ignored. This procedure reduced the features to only 5 out of 23. Table (4.13) presents the feature weights according to the FM.

Table 4.13: The Relevant Features of Sentimental LIAR According to FM

Features	Description	weight
Speaker_job	The job of the speaker	0.349
Subject	The type of topic, for example, political, economic, etc	0.275
Sentiment_sad	The speaker's feeling of sadness	0.250
Sentiment	Feeling of the speaker in general	0.250
Sentiment_joy	The speaker's feeling of joy	0.240

From Table (4.13), it can be concluded that the relevant speaker-based features that can be considered when checking a news article's reliability are Sentiment_sad, Sentiment_joy, Sentiment, Subject, and Speaker_job.

4.7 Results of Developed Machine Learning Model

The DMLM is proposed to improve the prediction of Arabic and English fake news. The proposed model focuses on one of the most important machine learning algorithms, the random forest classifier, and modifies its behaviour in building decision trees. Instead of randomly selecting the features in each sample, we have made the random forest classifier focus on the relevant user-based features in its decisions. Thus, the user-based features defined by the FM are selected in all decision trees, while the features extracted by TF-IDF or BERT are randomly selected.

4.7.1 Experiments on Twitter Dataset

To build a reliable model on this dataset, the random forest model is improved by controlling the process of selecting features in each tree instead of random selection. First, the top 9 features identified by the FM shown in Table (4.9) were combined with the features extracted from the tweet itself by TF-IDF. Then, the indices of the nine user-based features is memorized (we specified the index for each one). In this dissertation, the relevant user-based features are initially concatenated with features extracted from TF-IDF (i.e., the indices of these features is from 0 to 8, as shown in Table (4.14).

Finally, the number of decision trees in the random forest classifier was initialized to be 100, and each tree requires 12 features in the construction process. Therefore, a random selection is made for TF-IDF features and the last four features identified by the fuzzy model. Meanwhile, the first five user-based features were selected for all trees. In other words, each tree of the random forest model is constructed from the five user-based features and ieght randomly selected features from the other features.

.

Table 4.14: Sample of Updated Twitter Dataset

$Feature_1$ has_user_mention	……	$Feature_9$ count_of_unique_words	$Feature_1$ النظام	………
0	……	0.1	0	………
0	……	0	0	………
1	……	0.15	0	………
0	……	0.23	0.294	………
1	……	0	0	………
-----	-----	------	------	-----

Our proposed model was trained based on the K-fold Cross-Validation approach. The 10-fold Cross-Validation method is adopted for training and testing the proposed model and baseline model (LR). This means that the Twitter dataset was divided into ten parts, nine of which are used for training, and one part is used for testing. This process is repeated ten times; thus, the model's accuracy is calculated by taking the average.

The Twitter dataset (a sample of which is shown in Table 4.14) was expanded and updated by adding approximately 1,138 tweets and labeled manually by the expert. Thus, the total number of tweets in the updated Twitter dataset equals 3,000 tweets. Based on the Twitter dataset and the updated Twitter dataset, the quality of the proposed model was evaluated. The common evaluation metrics used in this dissertation are accuracy, precision, recall, and F-measure. Table (4.15) presents the results of our proposed model, the baseline model, and the results of previous studies conducted on the Twitter dataset.

Table (4.15): The Results of the Proposed Model (DMLM) Based on the Twitter Dataset

Dataset	The Model		Accuracy	Precision	Recall	F-Measure
Twitter Dataset (User-Based Features)	RF		0.747	0.485	0.747	0.748
	LR		0.722	0.431	0.720	0.723
Feature Extracted by TF-IDF (Original)	RF		0.799	0.615	0.803	0.798
	LR		0.747	0.485	0.747	0.748
Expanded dataset (from 1862 to 3000 instances)	RF		0.849	0.695	0.850	0.850
	LR		0.750	0.505	0.759	0.740
The Final Updated Dataset (Top Feature and TF-IDF)	**DMLM**		**0.889**	**0.881**	**0.891**	**0.884**
	LR		0.775	0.591	0.775	0.772
Twitter Dataset (User-Based Features)	The Study in [1]	RF	0.79	0.76	0.83	----
Twitter Dataset (NLP)	The Study in [2]	RF	0.7539	0.7603	0.8240	0.7909
		VBHHO	0.8150	0.8235	0.8552	0.8390

Table (4.15) shows the possible outcomes of predictive models when they are trained on user-based features or features extracted from the tweet using the TF-IDF method. This table also presents the results of previous studies conducted on the Twitter dataset. This dissertation is concerned with the development of the random forest classifier, so the performance of this classifier will be monitored by taking the accuracy criterion as an example from the above table. The random forest classifier, when it was trained on the features extracted by the TF-IDF method, achieved an accuracy of 0.799, and thus it performed better when it was trained on the user-based features.

When the dataset was expanded from 1862 to 3000, the random forest classifier achieved an accuracy of 0.849, and thus the model improved 5.8%. The final stage is to merge the important features identified by the FM with the features extracted from the TF-IDF method. Our proposed model was trained on the updated dataset and achieved an accuracy equal to 0.889. In comparison with previous studies conducted on this dataset (the studies by [1] and [2]) and the LR that was used as a baseline model, Figure (4.7) shows the preference of our proposed model in terms of accuracy.

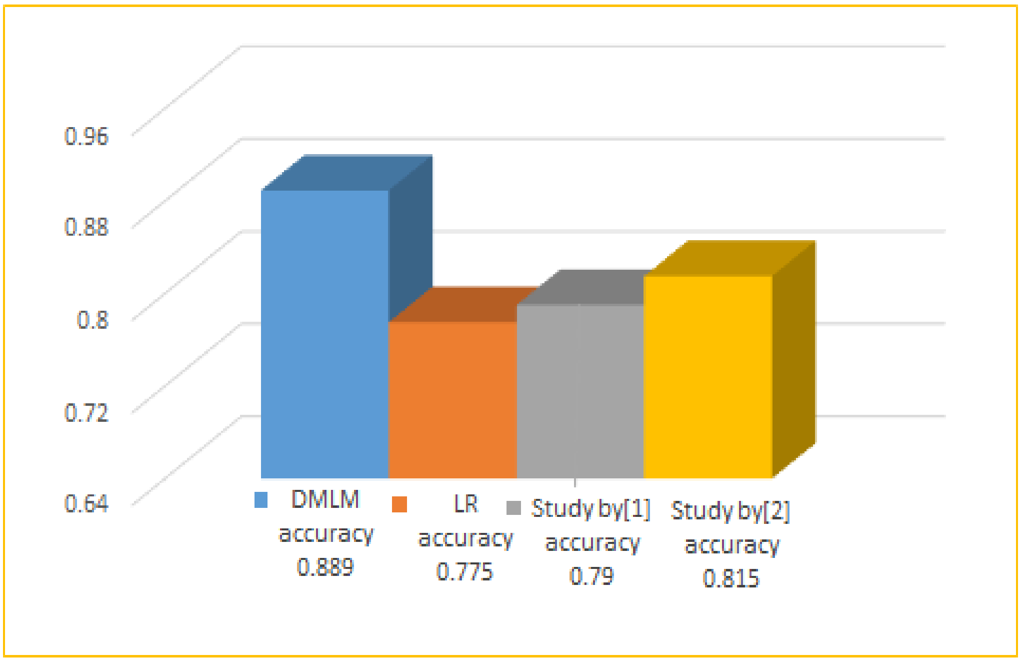


Figure (4.7): The Proposed Model Accuracy of the Twitter Dataset

Overall, our proposed method outperformed the LR and previous studies (the studies by [1] and [2]) in terms of accuracy and other evaluation metrics. Moreover, the improvement ratio between the proposed and LR models was 12.52%. The improvement ratio between the proposed model and the best accuracy obtained in previous studies is 8.013%. Finally, the evaluation metrics of our proposed model and other models are shown in Figure (4.8).

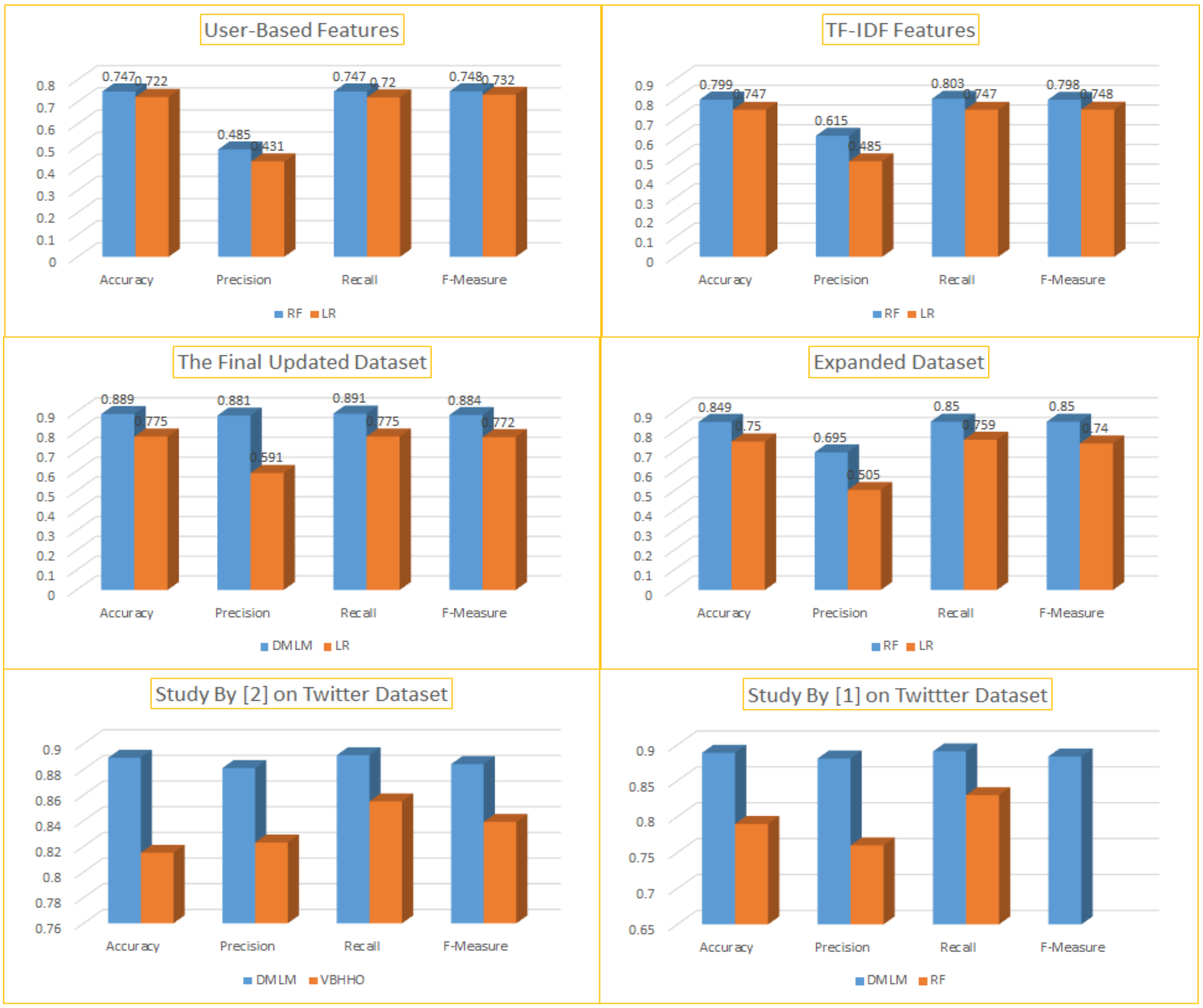


Figure (4.8): The Evaluation Metrics for Twitter Datasets

4.7.2 Experiments on Sentimental LAIR Dataset

The DMLM was also tested on the Sentimental LIAR dataset. Each record of this dataset contains a sentence (text) and a set of speaker-based features. Applying the FM for feature selection identified the five best features among all the features (see Table 4.13). As described in the feature extraction section, the BERT pre-trained model is used for word representation. So, the first procedure combines the features defined by the FM with the features obtained by the BERT pre-trained model. After that, the relevant speaker-based features were located, so their index would be from 0 to 4, as shown in Table 4.16.

Table 4.16: Sample of Updated Sentimental LIAR Dataset

Feature $_1$ sentiment_sad	……	*Feature* $_5$ speaker_job	*Feature* $_6$ say	………
-0.53925		Senator	-0.21308	………
-0.18179		Senator	-0.26108	………
-0.16628		Governor	0.007978	………
-0.39333		Senator	-0.24106	………
0.044449		President	-0.06197	………
-------		-------	------	-------

In our proposed model, the number of trees was set to 100 trees in the forest, and the number of features in each tree was identified to be 10. Thus, in each decision tree in the model, the five features identified by the FM were selected, and the remaining five features were randomly selected from the features extracted by the BERT pre-trained model.

In the experiments, the 10-fold cross-validation method was used to train and test the proposed and baseline models. The performance of these models was identified by the evaluation metrics: accuracy, precision, recall, and f-measure. The results of our proposed model, the baseline model (RF and LR), and the previous study conducted on the Sentimental LIAR dataset are presented in Table (4.17) and Figure (4.9).

Table 4.17: The Results of DMLM (Proposed Model) Based on Sentimental LIAR Dataset

The Model	Accuracy	Precision	Recall	F-Score
DMLM	**0.762**	**0.782**	**0.695**	**0.736**
RF	**0.729**	**0.735**	**0.730**	**0.723**
LR	0.651	0.620	0.624	0.647
The Study in [7]	0.700	----	----	0.6370

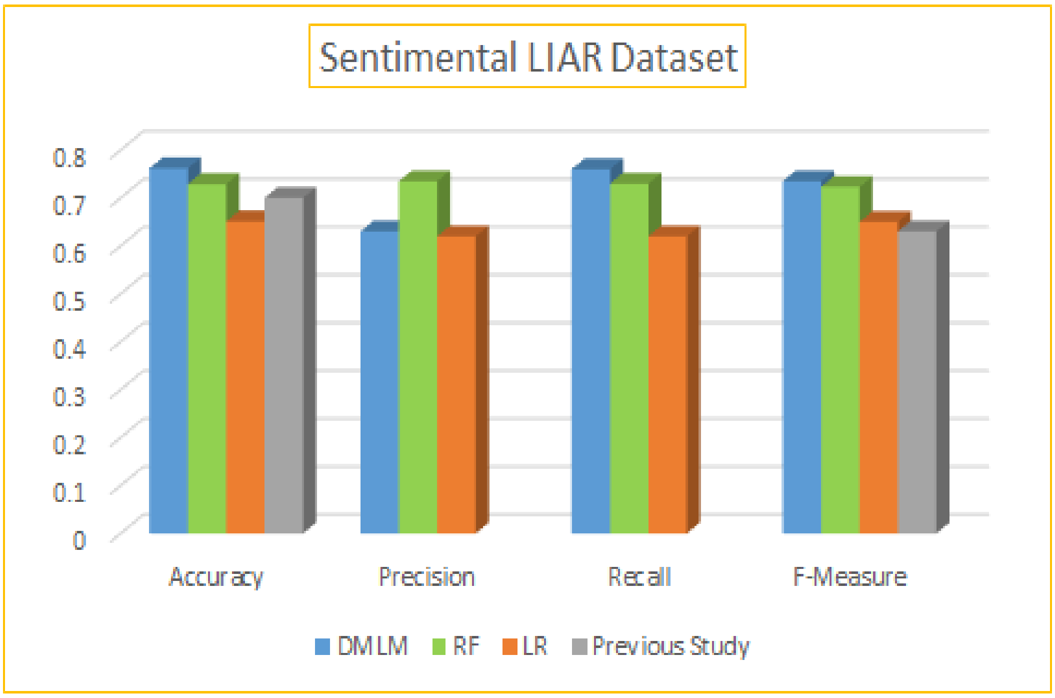


Figure (4.9): The Evaluation Metrics for Sentimental LIAR

According to Table (4.17) and in comparison with the previous study conducted on the Sentimental LIAR dataset, our proposed model is superior in prediction accuracy, precision, recall, and f-measure. It also outperformed the LR that was used as a baseline model. In addition, the ratio of accuracy improvement between our proposed model and the model proposed in the previous study is 8.13%, while this improvement in comparison with the RF and LR is 4.33% and14.57% respectively.

4.8 Results of Proposed Deep Learning Models

This dissertation proposes two models, ABHDM and EDLM, to improve the prediction of fake news in English and Arabic. Both models were built based on deep learning techniques and have achieved promising results in dealing with long and short news articles. The experiments were conducted using the AFND, the AraNews dataset, and the Fake-or-Real dataset, which were explained in the previous sections.

ABHDM is a combination of CNN, Bi-LSTM, Bi-GRU, and the Attention model. The idea behind hybridizing these networks into a single model is to contextualize long-term dependencies by using the Bi-LSTM and the Bi-GRU layers and reducing the dimensionality of the dataset and identifying the most informative words through the use of the convolution layer, max pooling layer, and attention layers. In contrast, the EDLM builds its predictions by adapting the common ensemble approach in machine learning to deep learning. Thus, any news article is classified as fake or real by taking the opinion of three deep learning techniques.

In both models, the following procedures were used: (1) the Holdout technique is used to train and test the models, where all datasets were divided into 80% for training and 20% for testing; (2) the accuracy metric is used to identify the convergence of the proposed models and binary cross-entropy is utilized as the loss function; (3) the Adam optimizer is used to update hyperparameters in back-propagation; (4) Keras is used to implement these deep models.

The ABHDM and EDLM are optimized in three ways: early stopping, dropout, and batch normalization. Early stopping based on accuracy with patience of 4 is used to avoid excessive training. The dropout layer with a probability of 0.25 is used to avoid overfitting. To accelerate the network training, batch normalization is employed.

On the other hand, hyperparameters play an important role in network training convergence and non-fall in local minima. As such, the batch size is set as 256. The learning and decay rates were set as 10^{-3} and $10^{-10,}$ respectively. Finally, the number of epochs was determined by trial and error to be 25 (this challenge was addressed by using early stopping). Table (4.18) summarises the hyperparameter's setting of the proposed models.

Table (4.18). The Hyperparameters Setting of Proposed Models

Parameter	Size
Training Set	80%
Testing Set	20%
Batch Size	256
Dropout Rate	0.25
Learning Rate	10^{-3}
Decay Rate	10^{-10}
Patience	4
Epochs	15, 20, and early stopping

4.8.1 Experiments on AraNews Dataset

Both proposed models were trained and tested on the AraNews dataset after preprocessing and preparing it to be in an acceptable format when introduced to deep learning techniques (see Sections 4.4 and 4.5). The common layers in the proposed models are the embedding layer, the convolution layer, the Bi-LSTM layer, and the Bi-GRU layer. Thus, the parameters of these layers and other hyperparameters have been set because this step is very important to improve the accuracy of models.

To build the input matrix in the embedding layer, 7000 tokens (words) are considered in the tokenization stage. By defining the padding size to 80, the first 80 words in each news article are considered (i.e., max length equal to 80). In contrast, any news article of fewer than 80 words will be padded with zeros. This layer (embedding layer) is concerned with word representation, so it is usually used to perform two tasks:

The first task is the possibility of loading the embedding matrix resulting from the pre-trained models, and in this case, it is prepared as untrainable.

The second task of this layer is to randomly generate an embedding matrix and make it trainable so that this matrix is updated as the network is trained.

Thus, the output of the pre-trained FastText word embedding model with an embedding dimension equal to 300 was loaded to this layer. In contrast, when the embedding layer itself is used for word representation (not using pre-trained models), the embedding dimension is set to 150.

In CNN, only two layers are added: the convolution layer and the max pooling layer. The kernel size of 3 with 32 filters is utilized in the convolutional layer. In this layer, the ReLU activation function was applied to introduce the output. On the output of this layer, the max pooling layer is added to identify relevant features. Bi-GRU and Bi-LSTM, each with 64 neurons, are utilized to handle short and longer sequences. In both these layers, ReLU is used. The summary of parameters tunning was presented in Table (4.19).

Table (4.19). The Summary of Parameters Setting of AraNews Dataset

Name	Size
Max Features	7000
Embedding Dim.	300, 150
Max-Length	80
Number of Filters	32
Kernel Size	3
No. of Units	64

To prove the effectiveness of ABHDM and EDLM, they were compared with three previous studies that were conducted on the AraNews Dataset (the studies by [3], [4], and [5]). Bi-LSTM, Bi-GRU, and CNN are also used for comparison because these networks contributed to building our proposed models , so they trained individually. Moreover, the proposed models' results are better than all these models, as shown in Table (4.20) and Figure (4.10).

Table (4.20): The Results and Comparisons of the ABHDM and EDLM Based on the AraNews Dataset Without Early Stopping Technique

The Model	Word Rep. Tech.	Epochs	Accuracy	Precision	Recall	F-Score	Time Per Epoch
CNN	FastText	25	0.8984	0.8891	0.8815	0.8853	17s
Bi-LSTM	FastText	50	0.8714	0.9022	0.7972	0.8465	23s
Bi-GRU	FastText	50	0.8349	0.8572	0.7544	0.8025	21s
ABHDM	**FastText**	**20**	**0.9359**	**0.9402**	**0.9283**	**0.9317**	**26s**
EDLM	**FastText**	**20**	**0.9032**	**0.9001**	**0.8897**	**0.8973**	**73s**
The Study in [3]	----	----	0.866	----	----	----	----
The Study in [4]	----	----	0.8002	0.8004	0.9991	----	----
The Study in [5]	----	----	0.80	----	----	----	----
ABHDM	Keras Emb. Layer	25	0.9205	0.9299	0.9137	0.9251	31s
EDLM	Keras Emb. Layer	25	0.8990	0.8789	0.9095	0.8881	85s

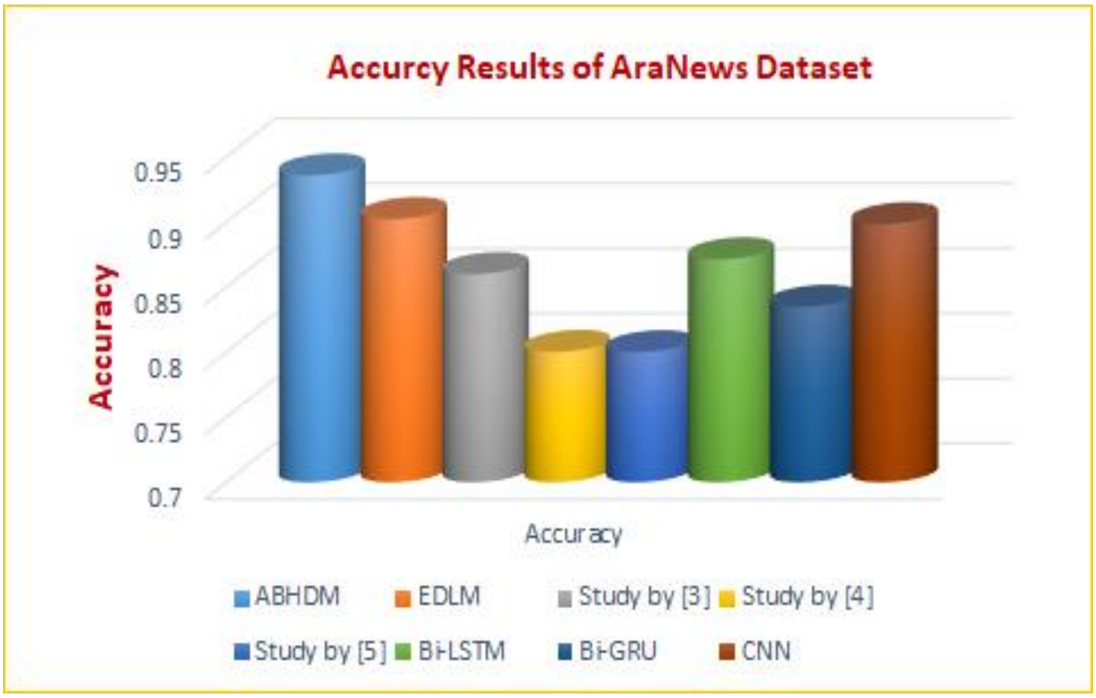


Figure (4.10): The Accuracy of AraNews Dataset

According to Table (4.20), it was found that using pre-trained models to represent words achieves better prediction accuracy and execution time than using the Keras embedding layer to represent words. On the other hand, our proposed models have achieved better results than the baseline models and previous studies that used the same dataset. As such, the improvement ratio of our proposed models compared to other models and studies can be summarized as follows:

(1) the improvement ratios are 4.06%, 6.89%, 10.79%, 7.46%, 14.52%, and 14.52% in comparison between the ABHDM and the CNN, BiLSTM, Bi-GRU, study by [3], study by [4], and study by [5] respectively.

(2) the improvement ratio is 1.01%, 3.52%, 7.56%, 4.11%, 11.42%, and 11.42% in comparison between the EDLM and the CNN, BiLSTM, Bi-GRU, study by [3], study by [4], and study by [5] respectively.

Finally, Figures (4.11) and (4.12) explain the convergence of AMHDM and EDLM in terms of accuracy and loss function, respectively, whereas Figures (4.13) and (4.14) show the confusion matrix of these models.

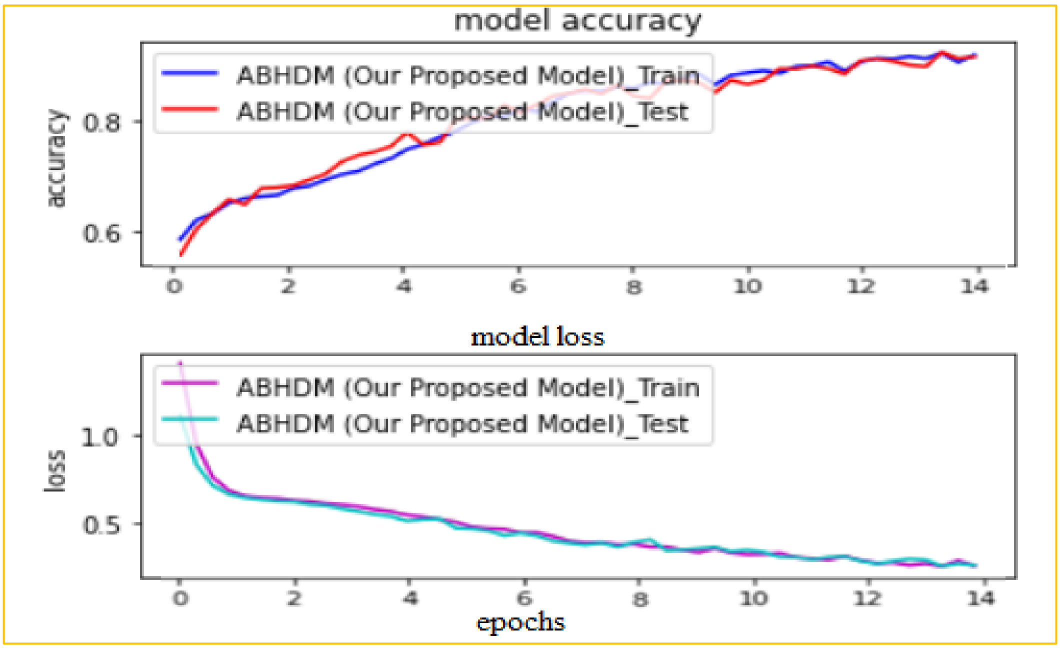


Figure (4.11): The ABHDM Convergence of AraNews Dataset

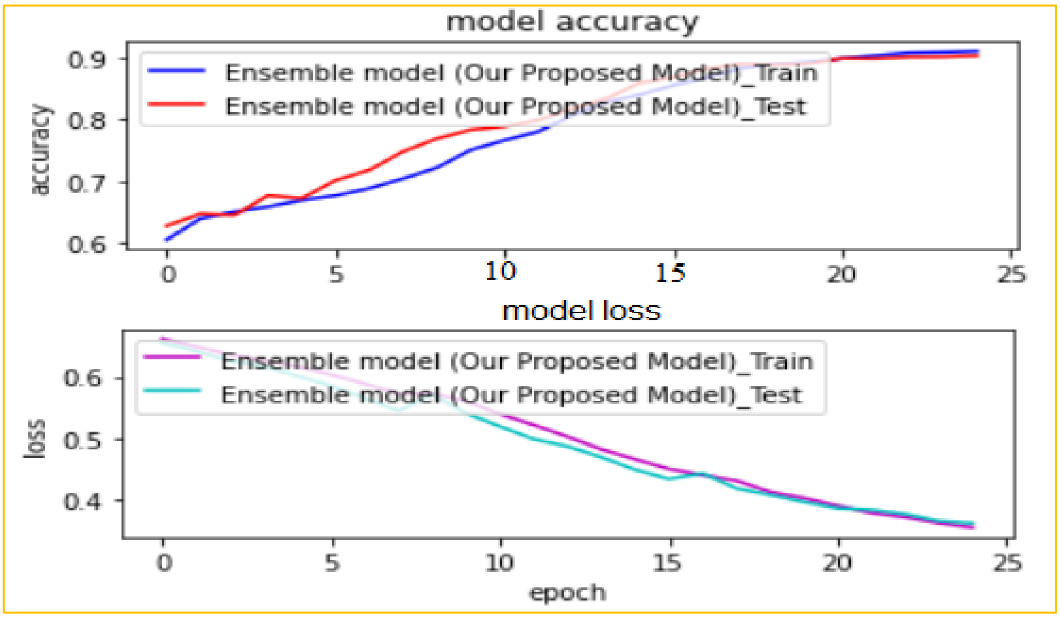


Figure (4.12): The EDLM Convergence of AraNews Dataset

	Confusion Matrix / ABHDM AraNews Dataset	
Not-Fake	**1434**	**89**
Fake	**124**	**1685**

Figure (4.13): The ABHDM Confusion Matrix of AraNews Dataset

	Confusion Matrix / EDLM AraNews Dataset	
Not-Fake	**1359**	**151**
Fake	**171**	**1634**

Figure (4.14): The EDLM Confusion Matrix of AraNews Dataset

The results of fake news predictions improved when using the early-stopping technique. Table (4.21) presents the prediction results after the early stopping technique was used instead of the epochs parameter.

Table (4.21): The Results of ABHDM and EDLM Based on the AraNews Dataset with Early Stopping and Batch Normalization

The Model	Early Stopping	Accuracy	Precision	Recall	F-Score	Time Per Epoch
ABHDM	35	0.9447	0.9609	0.9129	0.9363	24s
EDLM	41	0.9146	0.8561	0.9557	0.9032	63s

According to Table (4.21), the early stopping technique slightly improved the accuracy of both the ABHDM and the EDLM. Moreover, the improvement ratio of ABHDM accuracy is 1.02% with and without early stopping. In comparison, the improvement ratio of EDLM accuracy is 1.59% by using early stopping instead of defining the network with epochs by the user.

4.8.2 Experiments on AFND Dataset

AFND is a huge dataset and most of its news articles are lengthy, so the parameters of ABHDM and EDLM were set according to the nature of this dataset. The important layers of these models are configured as follows:

Initially, the embedding matrix generated by the FastText pre-trained model was loaded into the embedding layer to represent the words. This pre-trained model assigns a vector to each word based on its semantic meaning. To determine the dimensions of this embedding matrix, the max-words (max features) were determined to be 10,000 in size, while the emb_dim is 300, which the FastText pre-trained model specifies. The max_len of each article was identified as 128 to ensure that the models receive a fixed length of all articles. After that, the concept of truncating is applied to each news article that exceeds 128 words, and similarly, the padding with zeros is performed for news articles less than 128 words thus, the output of this layer is a matrix of size 10000* 300.

In the convolutional layer, a kernel size of 3 with 64 filters is utilized to reduce the feature space. Bi-GRU and Bi-LSTM, each with 64 neurons, address long-short dependencies. The ReLU activation function was applied in these three layers. Table (4.22) presents the parameters tunning of the proposed models.

Table (4.22). The Summary of Parameters tunning of AFND

Name	Size
Max Features	10000
Embedding Dim.	300, 150
Max-Length	128
Number of Filters	64
Kernel Size	3
No. of Units	64

So far, the proposed models have become trainable, so they were trained and their quality was determined using evaluation metrics. Based on the accuracy metric, the proposed models were compared with previous works and baseline models (CNN, Bi-LSTM, and Bi-GRU). Not many studies were conducted on the AFND. However, compared to the study by [6] that used the same dataset, it was found that the proposed models are better in terms of prediction accuracy. Table (4.23) presents the results of the proposed models and their comparison with baseline models and previous studies.

Table 4.23: The Results and Comparisons of the ABHDM and EDLM Based on AFND

The Model	Epochs	Accuracy	Precision	Recall	F-Score	Time Per Epoch
CNN	25	0.8109	0.8107	0.8047	0.8077	21s
Bi-LSTM	25	0.8121	0.7748	0.8172	0.8101	25s
Bi-GRU	25	0.8068	0.8120	0.7918	0.8018	23s
ABHDM	**15**	**0.8486**	**0.8492**	**0.8580**	**0.8516**	**35s**
EDLM	**20**	**0.8301**	**0.8219**	**0.8454**	**0.8308**	**77s**
The Study in [6]	----	0.79	----	----	----	

By looking at Table (4.23), it was found that the proposed models outperformed the baseline models and the previous study that used the AFND. Moreover, the improvement ratio in accuracy achieved through the proposed models can be explained as follows:

(1) the improvement ratio is 4.44%, 4.02%, 5.72%, and 6.90% in comparison between the ABHDM and the CNN, Bi-LSTM, Bi-GRU, and study by [6] respectively.

(2) the improvement ratio is 2.28%, 2.11%, 3.48%, and 4.96% in comparison between the EDLM and the CNN, BiLSTM, Bi-GRU, and study by [6] respectively.

Figure (4.15) shows the accuracy percentages obtained by the proposed models compared to the baseline models and the previous study.

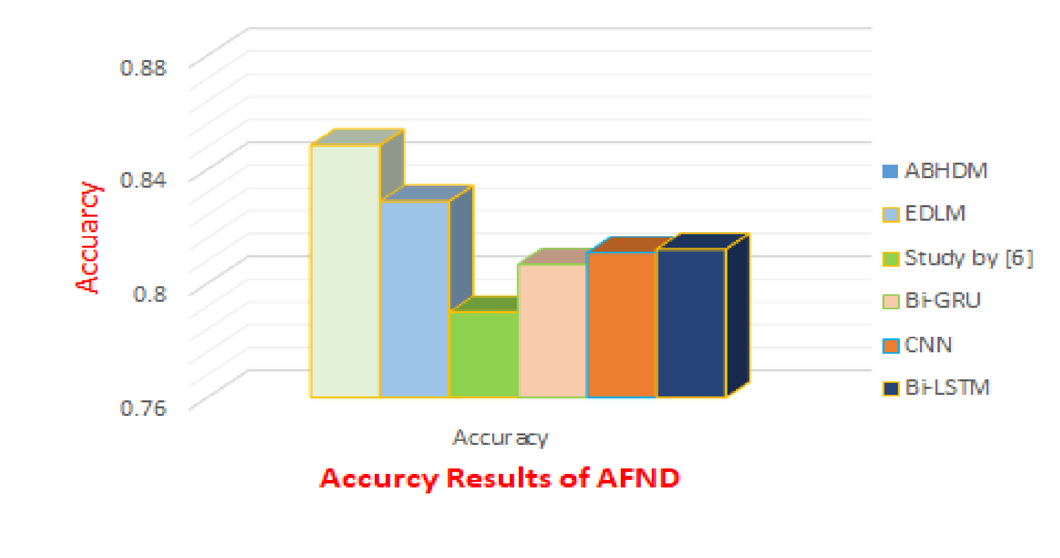


Figure (4.15): The Proposed Model Accuracy of AFND

Figures (4.16) and (4.17) show the convergence in training accuracy and test accuracy as well as the loss function of the proposed models, while Figures (4.18) and (4.19) present the confusion matrix for them.

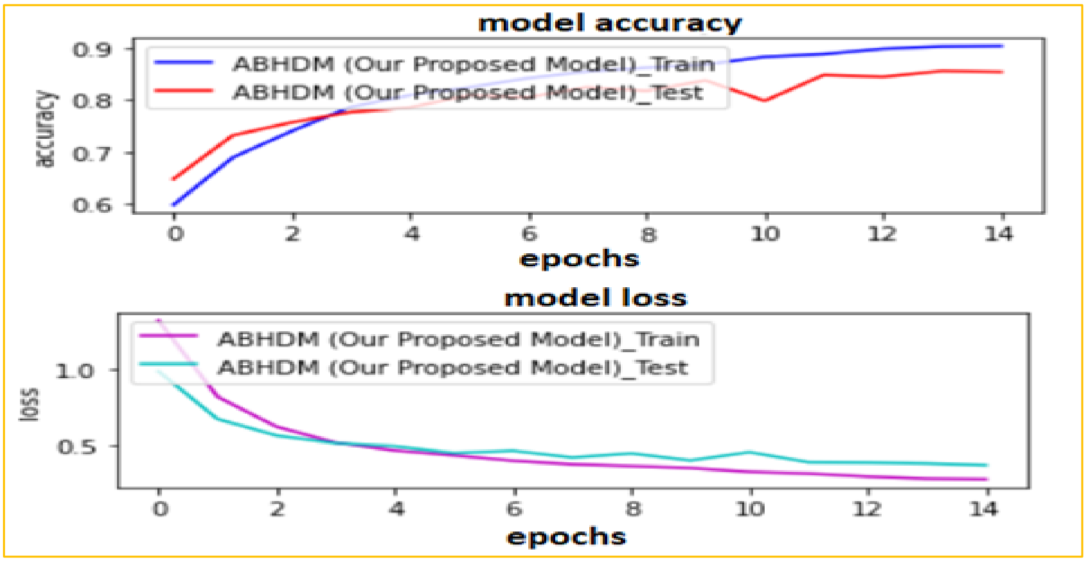


Figure (4.16): The ABHDM Convergence of AFND

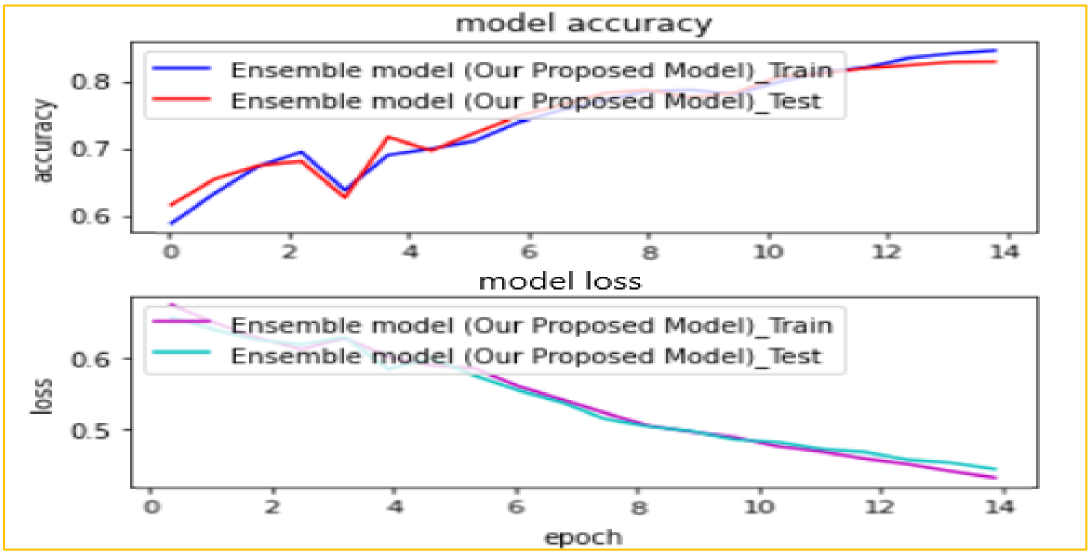


Figure (4.17): The EDLM Convergence of AFND

	Confusion Matrix / ABHDM AFND	
Not-Fake	**3531**	**627**
Fake	**584**	**3252**

Figure (4.18): The ABHDM Confusion Matrix of AFND

	Confusion Matrix / EDLM AFND	
Not-Fake	**3402**	**737**
Fake	**622**	**3237**

Figure (4.19): The EDLM Confusion Matrix of AFND

The proposed Models are optimized by applying early stopping and batch normalization. The fake news prediction accuracy of the proposed models improved by applying early stopping while the training time was reduced by using the batch normalization layer as shown in Table (4.24).

Table (4.24): The Results of the ABHDM and the EDLM Based on the AFND with Early Stopping and batch normalization

The Model	**Early Stopping**	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Time Per Epoch**
ABHDM	38	0.8595	0.8517	0.8572	0.8545	31s
EDLM	32	0.8363	0.8146	0.8581	0.8358	71s

According to Table (4.24), the accuracy improvement ratio for the ABHDM increased by 2.48% compared to what it was previously, while this improvement ratio amounted to 1.94% in the EDLM.

4.8.3 Experiments on Fake-or-Real Dataset

This dissertation is also concerned with English fake news detection by applying the proposed models to the Fake-or-Real Dataset. In both models, the GloVe pre-trained model was used for word representation. Then the proposed models were trained on this dataset after setting the parameters as shown in Table (4.25).

Table (4.25). The Summary of Parameters tunning of Fake-or-Real Dataset

Name	Size
Max Features	6000
Embedding Dim.	300, 150
Max-Length	100
Number of Filters	32
Kernel Size	3
No. of Units	64

After that, the proposed models were trained, and their performance was measured according to the evaluation metrics. Our results were compared with the baseline models and previous studies (study by [8]) conducted on this dataset. Moreover, the proposed models achieved the best results, as shown in Table (4.26).

Table (4.26): The Results and Comparisons of our Proposed Model Based on Fake-or-Real Dataset

The Model	**Epochs**	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Time Per Epoch**
CNN	20	0.9807	0.9742	0.9855	0.9798	19s
Bi-LSTM	25	0.9749	0.9817	0.9871	0.9842	24s
Bi-GRU	25	0.9708	0.9618	0.9774	0.9665	20s
ABHDM	**20**	**0.9944**	**0.9933**	**0.9953**	**0.9937**	**31s**
EDLM	**20**	**0.9881**	**0.9870**	**0.9892**	**0.9875**	**59s**
The Study in [8])	----	0.92	----	----	----	

Dealing with the English language in FND is less challenging than in Arabic. Therefore, the results shown in Table (4.26) are close, despite the superiority of our proposed models by a slight difference. Figure (4.20) presents the prediction accuracy ratios obtained by conducting experiments on the Fake-or-Real Dataset, while Figures (4.21) and (4.22) show the confusion matrix of ABHDM and EDLM.

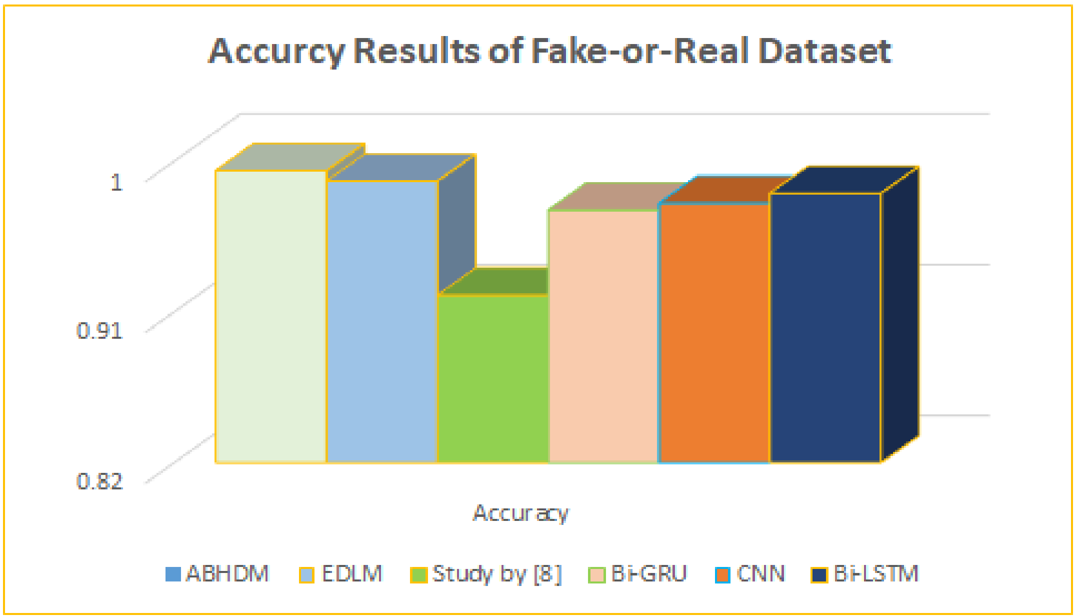


Figure (4.20): The Accuracy of Fake-or-Result Dataset

	Confusion Matrix / ABHDM Fake-or-Real Dataset	
Not-Fake	4476	30
Fake	21	4457

Figure (4.21): The ABHDM Confusion Matrix of Fake-or-Real

	Confusion Matrix / EDLM Fake-or-Real Dataset	
Not-Fake	4406	58
Fake	48	4032

Figure (4.22): The EDLM Confusion Matrix Fake-or-Real

4.9 Analysis of ABHDM Layers and Justification

Both proposed models, ABHDM and EDLM, are hybrid models from deep neural networks. For the EDLM, the CNN, the Bi-LSTM, and the Bi-GRU are complete networks, each of which decides the credibility of the news article. Then, these decisions are collected in order to classify any news article. This concept is implied under the ensemble approach, a common machine learning mechanism. This dissertation focused on simulating this common approach in machine learning and its application in deep learning. When training individually, the proposed model proved its superiority compared to CNN, Bi-LSTM, and Bi-GRU. As for the ABHDM, the mechanism is somewhat different, as this dissertation is concerned with adding one or two layers from each network. Each layer performs a specific task, and the contributions of each layer can be explained as follows:

1. Convolution and Max Pooling Layers (CNN): CNN is characterized by its ability to deal well with big data and feature selection. Therefore, convolution and max pooling layers were used in ABHDM to reduce the dataset's dimensionality and identify the relevant features.
2. Bi-LSTM Layer and Bi-GRU Layer: In the proposed model, a single layer of B--LSTM and Bi-GRU networks was used because it efficiently handles long and short-term dependencies (processing long and short sequences of texts).
3. Attention Layer: The goal of using the attention layer is to generate a context vector representing the context of the entered sentence.
4. Output Layer (MLP): The final stage in addressing any news article is to classify it as real or fake by using output layer.

This dissertation employed the above five layers and other layers, such as the dropout and normalization layers, to speed up network training and improve prediction accuracy in building a robust and effective model for

FND. The contributions of each layer to improving the accuracy of fake news prediction are shown in the four points above. In fact, this dissertation focuses on knowing the effect of each layer of the ABHDM in improving accuracy. Therefore, four possible hybrid architectures were constructed from this model: CNN_Bi-LSTM_MLP, CNN_Bi-GRU_MLP, Bi-LSTM_Attention_MLP, and Bi-GRU_Attention_MLP.

The purpose of building these hybrid models is to seek the possibility of obtaining a prediction accuracy equal to or close to the prediction accuracy in the ABHDM with the least number of layers. However, by training these hybrid networks and testing them in detecting fake news, it was found that the proposed model achieved better prediction accuracy, as shown in Table (4.27). Finally, and through the practical experiments of this dissertation, we must note that it is not necessary that adding more layers to the network leads to more complexity in terms of training time. The reason for this is that some layers are interested in reducing the dimensionality of the data and thus may facilitate the actions that must be taken in the next layer, which speeds up the training of the network (e.g., max pooling layers and batch normalization layer).

Table (4.27): The Results of Hybrid Models Based on the two Dataset

Dataset	The Model	Epochs (Early Stopping)	Accuracy	Time Per Epoch
AraNews	CNN_Bi-LSTM_MLP	63	0.9200	32s
	CNN_Bi-GRU_MLP	58	0.9191	29s
	Bi-LSTM_Attention_MLP	48	0.9108	25s
	Bi-GRU_Attention_MLP	35	0.9232	22s
AFND	CNN_Bi-LSTM_MLP	43	0.8150	47s
	CNN_Bi-GRU_MLP	48	0.8232	41s
	Bi-LSTM_Attention_MLP	41	0.8132	33s
	Bi-GRU_Attention_MLP	39	0.8201	28s

Chapter Five

Conclusions and Future Works

Conclusions and Future Works

5.1 Conclusion

In the last few years, it was observed that the data circulated via the Internet and social media increases dramatically, and thus transferring fake news within such data and misleading people is easy for the user or publisher. The research and academic community have given a significant interest in mitigating the consequences of spreading fake news through the use of ML and DL. The proposed system aims to improve the prediction accuracy of Arab fake news circulating on social media or other sources by focusing on relevant user-based and content-based features. Relevant features are selected through a technique of feature selection developed using fuzzy logic. Next, the random forest algorithm was improved by controlling the process of selecting features in each tree for Arabic fake news detection.

Nowadays, the modern trend is to use DL techniques, especially BiLSTM and BiGRU in NLP and FND. These existing networks are characterized by their high performance, but at the same time, they have some drawbacks in that it gives equal importance to all the words in the sentence. In this dissertation, two new deep neural networks are proposed, ABHDM and EDLM, to address these drawbacks through the use of the CNN and attention mechanism to focus on the more informative words.

Based on the findings of the proposed models, many conclusions can be drawn:

1. Performing data preprocessing especially, removing the missing rows, meaningless sentences, and noise led to better training (execution time) and prediction accuracy.
2. The use of pre-trained models and loading their results into the embedding layer improve model performance by assigning perfect

word representation and also improve time complexity by making this layer untrainable.

3. The proposed FM proved to be effective in selecting the relevant user-based features and content-based features and removing irrelevant ones or those that have negative effects. According to FM, it was found that the reduction rate of selection is equal to 78% for the features in the Twitter dataset and 81% for features in the Sentimental LIAR dataset. Moreover, it was concluded that some features such as "count of followers" has a greater impact (higher weights) on the model in the Twitter dataset. On the other hand, some features such as "speaker job" has a greater effect on the model in the Sentimental LIAR dataset.

4. The proposed DMLM by this dissertation has proved its efficiency in classifying Arabic fake news by improving prediction accuracy. Also, when the Twitter dataset was expanded, the model performance has been significantly improved compared to other studies.

5. In view of the fact that the ABHDM is a hybrid of two layers of the CNN, the Bi-LSTM layer, the Bi-GRU layer, and the attention layer, it achieved the best performance compared to previous studies and individual training of these networks. Likewise, the EDLM achieved higher accuracy than the networks contributing to its construction and previous studies.

6. By using the three optimization methods: early stopping, dropout, and batch normalization, the following was concluded:

    - The early stopping controlled the number of epochs, which increased the network training time, but in return, the prediction accuracy increased significantly.

- The time to update parameters and the network training is improved by 6 to 10 seconds per epoch through the use of a batch normalization layer.
- The dropout layer helped in the convergence between the training accuracy and the testing accuracy, and the absence of a high drop in the testing accuracy.

7. Finally, in the backpropagation stage, which in turn updates the network parameters, it was concluded that the Adam optimizer is the best among the other optimization methods.

5.2 Future Work

Some future work can be listed below:

1- Due to the limited data available in Arabic fake news detection, in future work, we can use data augmentation techniques, which in turn augment training data to improve the prediction accuracy.

2- Applying the proposed model to social media platforms like Twitter and Facebook online (real time).

3- Improve the proposed deep models by initializing them with optimal parameters instead of random initialization at the start of training (e.g., by using a genetic algorithm, we can tune network parameters).

4- Study other metaheuristic algorithms such as particle swarm optimization for optimizing the proposed deep models to update weights and biases and reduce model complexity.

5- This dissertation focused on fake news detection. In the future, we can use our proposed models to include other areas, such as sentiment analysis, detecting deceptive opinions, and recognizing stances in debates.

References

[1] G. Jardaneh, H. Abdelhaq, M. Buzz, and D. Johnson, "Classifying Arabic Tweets Based on Credibility Using Content and User Features," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Apr. 2019, doi: 10.1109/jeeit.2019.8717386.

[2] T. Thaher, M. Saheb, H. Turabieh, and H. Chantar, "Intelligent Detection of False Information in Arabic Tweets Utilizing Hybrid Harris Hawks Based Feature Selection and Machine Learning Models," Symmetry, vol. 13, no. 4, p. 556, Mar. 2021, doi: 10.3390/sym13040556.

[3] F. Aljwari, W. Alkaberi, A. Alshutayri, E. Aldhahri, N. Aljojo, and O. Abouola, "Multi-scale Machine Learning Prediction of the Spread of Arabic Online Fake News," Postmodern Openings, vol. 13, no. 1 Sup1, pp. 01–14, Mar. 2022, doi: 10.18662/po/13.1sup1/411.

[4] M. Al-Yahya, H. Al-Khalifa, H. Al-Baity, D. AlSaeed, and A. Essam, "Arabic Fake News Detection: Comparative Study of Neural Networks and Transformer-Based Approaches," Complexity, vol. 2021, pp. 1–10, Apr. 2021, doi: 10.1155/2021/5516945.

[5] W. Shishah, "JointBert for Detecting Arabic Fake News," IEEE Access, vol. 10, pp. 71951–71960, 2022, doi: 10.1109/access.2022.3185083.

[6] A. Khalil, M. Jarrah, M. Aldwairi, and Y. Jararweh, "Detecting arabic fake news using machine learning," in 2021 Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA), 2021, pp. 171–177.

[7] B. Upadhayay and V. Behzadan, "Sentimental LIAR: Extended Corpus and Deep Learning Models for Fake Claim Classification," 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Nov. 2020, doi: 10.1109/isi49825.2020.9280528.

[8] H. Ahmed, I. Traore, and S. Saad, "Detecting opinion spams and fake

news using text classification," Security and Privacy, vol. 1, no. 1, pp. 1-15, Dec. 2018.

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

[10] N. A. Hamzah and B. N. Dhannoon, "The Detection of Sexual Harassment and Chat Predators Using Artificial Neural Network," Karbala International Journal of Modern Science, vol. 7, no. 4, Dec. 2021, doi: 10.33640/2405-609x.3157.

[11] Hussien et al., "Anomaly Detection Approach Based on Deep Neural Network and Dropout," Baghdad Science Journal, vol. 17, no. 2(SI), p. 0701, Jun. 2020, doi: 10.21123/bsj.2020.17.2(si).0701.

[12] H. Himdi, G. Weir, F. Assiri, and H. Al-Barhamtoshy, "Arabic Fake News Detection Based on Textual Analysis," Arabian Journal for Science and Engineering, vol. 47, no. 8, pp. 10453–10469, Feb. 2022, doi: 10.1007/s13369-021-06449-y.

[13] G. Gravanis, A. Vakali, K. Diamantaras, and P. Karadais, "Behind the cues: A benchmarking study for fake news detection," Expert Systems with Applications, vol. 128, pp. 201–213, Aug. 2019, doi: 10.1016/j.eswa.2019.03.036.

[14] S. Gao et al., "Short-term runoff prediction with GRU and LSTM networks without requiring time step optimization during sample generation," Journal of Hydrology, vol. 589, p. 125188, Oct. 2020, doi: 10.1016/j.jhydrol.2020.125188.

[15] M. Al-Fawa'reh, Z. Ashi, and M. T. Jafar, "Detecting Malicious DNS Queries Over Encrypted Tunnels Using Statistical Analysis and Bi-Directional Recurrent Neural Networks," Karbala International Journal of Modern Science, vol. 7, no. 4, Dec. 2021, doi: 10.33640/2405-609x.3155.

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

[17] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis," Future Generation Computer Systems, vol. 115, pp. 279–294, Feb. 2021, doi: 10.1016/j.future.2020.08.005.

[18] J. Žižka, F. Dařena, and A. Svoboda, "Introduction to Text Mining with Machine Learning," Text Mining with Machine Learning, pp. 1–12, Oct. 2019, doi: 10.1201/9780429469275-1.

[19] D. Jain and V. Singh, "Feature selection and classification systems for chronic disease prediction: A review," Egyptian Informatics Journal, vol. 19, no. 3, pp. 179–189, Nov. 2018, doi: 10.1016/j.eij.2018.03.002.

[20] B. Remeseiro and V. Bolon-Canedo, "A review of feature selection methods in medical applications," Computers in Biology and Medicine, vol. 112, p. 103375, Sep. 2019, doi: 10.1016/j.compbiomed.2019.103375.

[21] P. Saqib, U. Qamar, A. Aslam, and A. Ahmad, "Hybrid of Filters and Genetic Algorithm - Random Forests Based Wrapper Approach for Feature Selection and Prediction," Intelligent Computing, pp. 190–199, 2019, doi: 10.1007/978-3-030-22868-2_15.

[22] R. Saidi, W. Bouaguel, and N. Essoussi, "Hybrid Feature Selection Method Based on the Genetic Algorithm and Pearson Correlation Coefficient," Studies in Computational Intelligence, pp. 3–24, Dec. 2018, doi: 10.1007/978-3-030-02357-7_1.

[23] M. Lu, "Embedded feature selection accounting for unknown data heterogeneity," Expert Systems with Applications, vol. 119, pp. 350–361, Apr. 2019, doi: 10.1016/j.eswa.2018.11.006.

[24] G. Ansari, T. Ahmad, and M. N. Doja, "Hybrid Filter–Wrapper

Feature Selection Method for Sentiment Classification," Arabian Journal for Science and Engineering, vol. 44, no. 11, pp. 9191–9208, Jul. 2019, doi: 10.1007/s13369-019-04064-6.

[25] G. Chaudhary, "Overview and Application of Text Data Pre-Processing Techniques for Text Mining on Health News Tweets," HELIX, vol. 8, no. 5, pp. 2764–2768, Aug. 2018, doi: 10.29042/2018-3764-3768.

[26] M. Arief and M. B. M. Deris, "Text Preprocessing Impact for Sentiment Classification in Product Review," 2021 Sixth International Conference on Informatics and Computing (ICIC), Nov. 2021, doi: 10.1109/icic54025.2021.9632884.

[27] F. P. Shah and V. Patel, "A review on feature selection and feature extraction for text classification," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Mar. 2016, doi: 10.1109/wispnet.2016.7566545.

[28] R. Dzisevic and D. Sesok, "Text Classification using Different Feature Extraction Approaches," 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Apr. 2019, doi: 10.1109/estream.2019.8732167.

[29] M. Zareapoor and S. K. R, "Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection," International Journal of Information Engineering and Electronic Business, vol. 7, no. 2, pp. 60–65, Mar. 2015, doi: 10.5815/ijieeb.2015.02.08.

[30] X. Wen, "Topological Data Analysis In Text Classification Based On Word Embedding And TF-IDF," Journal of Physics: Conference Series, vol. 1634, no. 1, p. 012039, Sep. 2020, doi: 10.1088/1742-6596/1634/1/012039.

[31] A. Ardiansyah, S. Aji, D. Pratmanto, S. J. Kuryanti, O. P. Putra, and

C. Adiwihardja, “The Analysis of Digital Provider Sentiment Of ‘by.u’ On Google Play Which Uses The Support Vector Machine (SVM) Method,” Journal of Physics: Conference Series, vol. 1641, no. 1, p. 012048, Nov. 2020, doi: 10.1088/1742-6596/1641/1/012048.

[32] Y. Li and T. Yang, “Word Embedding for Understanding Natural Language: A Survey,” Guide to Big Data Applications, pp. 83–104, May 2017, doi: 10.1007/978-3-319-53817-4_4.

[33] S. Ruder, I. Vulić, and A. Søgaard, “A Survey of Cross-lingual Word Embedding Models,” Journal of Artificial Intelligence Research, vol. 65, pp. 569–631, Aug. 2019, doi: 10.1613/jair.1.11640.

[34] Z. Parcheta, G. Sanchis-Trilles, F. Casacuberta, and R. Rendahl, “Combining Embeddings of Input Data for Text Classification,” Neural Processing Letters, vol. 53, no. 5, pp. 3123–3151, Aug. 2020, doi: 10.1007/s11063-020-10312-w.

[35] F. Jahara, O. Sharif, and M. M. Hoque, “Automatic Categorization of News Articles and Headlines Using Multi-layer Perceptron,” Lecture Notes in Networks and Systems, pp. 155–166, 2022, doi: 10.1007/978-3-030-93247-3_16.

[36] H. Ahmed Chowdhury, Md. A. Haque Imon, and Md. S. Islam, “A Comparative Analysis of Word Embedding Representations in Authorship Attribution of Bengali Literature,” 2018 21st International Conference of Computer and Information Technology (ICCIT), Dec. 2018, doi: 10.1109/iccitechn.2018.8631977.

[37] C. Xiao, X. Hu, Z. Liu, C. Tu, and M. Sun, “Lawformer: A pre-trained language model for Chinese legal long documents,” AI Open, vol. 2, pp. 79–84, 2021, doi: 10.1016/j.aiopen.2021.06.003.

[38] J. Suzuki and M. Nagata, “A Unified Learning Framework of Skip-Grams and Global Vectors,” Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International

Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, doi: 10.3115/v1/p15-2031.

[39] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, doi: 10.3115/v1/d14-1162.

[40] B. Athiwaratkun, A. Wilson, and A. Anandkumar, "Probabilistic FastText for Multi-Sense Word Embeddings," Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, doi: 10.18653/v1/p18-1001.

[41] I. Santos, N. Nedjah, and L. de Macedo Mourelle, "Sentiment analysis using convolutional neural network with fastText embeddings," 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Nov. 2017, doi: 10.1109/la-cci.2017.8285683.

[42] R. K. Kaliyar, "A Multi-layer Bidirectional Transformer Encoder for Pre-trained Word Embedding: A Survey of BERT," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Jan. 2020, doi: 10.1109/confluence47617.2020.9058044.

[43] J.C. Gu, C. Tao, Z. Ling, C. Xu, X. Geng, and D. Jiang, "MPC-BERT: A Pre-Trained Language Model for Multi-Party Conversation Understanding," Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, doi: 10.18653/v1/2021.acl-long.285.

[44] H. Choi, K. Cho, and Y. Bengio, "Fine-grained attention mechanism for neural machine translation," Neurocomputing, vol. 284, pp. 171–176, Apr. 2018, doi: 10.1016/j.neucom.2018.01.007.

[45] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism

of deep learning," Neurocomputing, vol. 452, pp. 48–62, Sep. 2021, doi: 10.1016/j.neucom.2021.03.091.

[46] D. Hu, "An Introductory Survey on Attention Mechanisms in NLP Problems," Intelligent Systems and Applications, pp. 432–448, Aug. 2019, doi: 10.1007/978-3-030-29513-4_31.

[47] M. Jain, A. Madeira, and M. A. Martins, "A Fuzzy Modal Logic for Fuzzy Transition Systems," Electronic Notes in Theoretical Computer Science, vol. 348, pp. 85–103, Mar. 2020, doi: 10.1016/j.entcs.2020.02.006.

[48] G. Bennouna and M. Tkiouat, "Fuzzy logic approach applied to credit scoring for microfinance in Morocco," Procedia Computer Science, vol. 127, pp. 274–283, 2018, doi: 10.1016/j.procs.2018.01.123.

[49] T. A. Wotaifi and E. S. Al-Shemary "Mining of Completion Rate of Higher Education Based on Fuzzy Feature Selection Model and Machine Learning Techniques," Nov. 2019, doi: 10.31219/osf.io/wjbfk.

[50] M. M. Mirończuk and J. Protasiewicz, "A recent overview of the state-of-the-art elements of text classification," Expert Systems with Applications, vol. 106, pp. 36–54, Sep. 2018, doi: 10.1016/j.eswa.2018.03.058.

[51] S. G. Burdisso, M. Errecalde, and M. Montes-y-Gómez, "A text classification framework for simple and effective early depression detection over social media streams," Expert Systems with Applications, vol. 133, pp. 182–197, Nov. 2019, doi: 10.1016/j.eswa.2019.05.023.

[52] N. Altman and M. Krzywinski, "Ensemble methods: bagging and random forests," Nature Methods, vol. 14, no. 10, pp. 933–934, Sep. 2017, doi: 10.1038/nmeth.4438.

[53] G. Stavropoulos, R. van Voorstenbosch, F.-J. van Schooten, and A. Smolinska, "Random Forest and Ensemble Methods," Comprehensive Chemometrics, pp. 661–672, 2020, doi: 10.1016/b978-0-12-409547-

2.14589-5.

[54] E. Y. Boateng and D. A. Abaye, “A Review of the Logistic Regression Model with Emphasis on Medical Research,” Journal of Data Analysis and Information Processing, vol. 07, no. 04, pp. 190–207, 2019, doi: 10.4236/jdaip.2019.74012.

[55] A. S. Tomar, M. Singh, G. Sharma, and K. V. Arya, “Traffic Management using Logistic Regression with Fuzzy Logic,” Procedia Computer Science, vol. 132, pp. 451–460, 2018, doi: 10.1016/j.procs.2018.05.159.

[56] H. Ramchoun, M. Amine, J. Idrissi, Y. Ghanou, and M. Ettaouil, “Multilayer Perceptron: Architecture Optimization and Training,” International Journal of Interactive Multimedia and Artificial Intelligence, vol. 4, no. 1, p. 26, 2016, doi: 10.9781/ijimai.2016.415.

[57] F. Arce, E. Zamora, G. Hernández, J. M. Antelis, and H. Sossa, “Recognizing Motor Imagery Tasks Using Deep Multi-Layer Perceptrons,” Lecture Notes in Computer Science, pp. 468–482, 2018, doi: 10.1007/978-3-319-96133-0_35.

[58] A. A. Heidari, H. Faris, S. Mirjalili, I. Aljarah, and M. Mafarja, “Ant Lion Optimizer: Theory, Literature Review, and Application in Multi-layer Perceptron Neural Networks,” Studies in Computational Intelligence, pp. 23–46, Feb. 2019, doi: 10.1007/978-3-030-12127-3_3.

[59] Q. Li et al., “A Survey on Text Classification: From Traditional to Deep Learning,” ACM Transactions on Intelligent Systems and Technology, vol. 13, no. 2, pp. 1–41, Apr. 2022, doi: 10.1145/3495162.

[60] A. Kumar, “A Study: Hate Speech and Offensive Language Detection in Textual Data by Using RNN, CNN, LSTM and BERT Model,” 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), May 2022, doi: 10.1109/iciccs53718.2022.9788347.

[61] B. Athiwaratkun and J. W. Stokes, “Malware classification with

LSTM and GRU language models and a character-level CNN," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar. 2017, doi: 10.1109/icassp.2017.7952603.

[62] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU," Journal of Artificial Intelligence and Soft Computing Research, vol. 9, no. 4, pp. 235–245, Aug. 2019, doi: 10.2478/jaiscr-2019-0006.

[63] S. K. DASARI and S. Mehta, "Scene Based Text Recognition and Classification Based on Hybrid Cnn Models with Performance Evaluation," SSRN Electronic Journal, 2022, pp. 1-17, doi: 10.2139/ssrn.4174796.

[64] J. Zhang, F. Liu, W. Xu, and H. Yu, "Feature Fusion Text Classification Model Combining CNN and BiGRU with Multi-Attention Mechanism," Future Internet, vol. 11, no. 11, p. 237, Nov. 2019, doi: 10.3390/fi11110237.

[65] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," 2018 Chinese Control And Decision Conference (CCDC), Jun. 2018, doi: 10.1109/ccdc.2018.8407425.

[66] J. Feng and S. Lu, "Performance Analysis of Various Activation Functions in Artificial Neural Networks," Journal of Physics: Conference Series, vol. 1237, no. 2, p. 022030, Jun. 2019, doi: 10.1088/1742-6596/1237/2/022030.

[67] M. Amrouche and D. M. Stipanovic, "A Formal Characterization of Activation Functions in Deep Neural Networks," IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14, 2022, doi: 10.1109/tnnls.2022.3187538.

[68] N. Li and X. Hou, "Research on Adaptive Quantum Forward Counter

Propagation Algorithm," Journal of Electronics & Information Technology, vol. 35, no. 11, pp. 2778–2783, Feb. 2014, doi: 10.3724/sp.j.1146.2013.00101.

[69] M. Fayaz, H. Shah, A. Aseere, W. Mashwani, and A. Shah, "A Framework for Prediction of Household Energy Consumption Using Feed Forward Back Propagation Neural Network," Technologies, vol. 7, no. 2, p. 30, Apr. 2019, doi: 10.3390/technologies7020030.

[70] M. R. Arefin and M. Asadujjaman, "Minimizing Average of Loss Functions Using Gradient Descent and Stochastic Gradient Descent," Dhaka University Journal of Science, vol. 64, no. 2, pp. 141–145, Jul. 2016, doi: 10.3329/dujs.v64i2.54490.

[71] S. Keydana, "Loss Functions," Deep Learning and Scientific Computing with R torch, pp. 75–82, Feb. 2023, doi: 10.1201/9781003275923-9.

[72] R. Malhotra, A. Shakya, R. Ranjan, and R. Banshi, "Software defect prediction using Binary Particle Swarm Optimization with Binary Cross Entropy as the fitness function," Journal of Physics: Conference Series, vol. 1767, no. 1, p. 012003, Feb. 2021, doi: 10.1088/1742-6596/1767/1/012003.

[73] V. Chahar, "Analysis of Back Propagation Algorithm," International Journal of Scientific Research, vol. 2, no. 8, pp. 305–306, Jun. 2012, doi: 10.15373/22778179/aug2013/98.

[74] Kim, "Development of Optimum Traffic Safety Evaluation Model Using the Back-Propagation Algorithm," Journal of the Korean Society of Civil Engineers, vol. 35, no. 3, p. 679, 2015, doi: 10.12652/ksce.2015.35.3.0679.

[75] R. Kong, P. Liu, and X. Zhou, "Multipass cell design with the random walk and gradient descent optimization algorithms," Applied Physics B, vol. 127, no. 9, Aug. 2021, doi: 10.1007/s00340-021-07679-6.

[76] A. Jentzen, B. Kuckuck, A. Neufeld, and P. von Wurstemberger, “Strong error analysis for stochastic gradient descent optimization algorithms,” IMA Journal of Numerical Analysis, vol. 41, no. 1, pp. 455–492, May 2020, doi: 10.1093/imanum/drz055.

[77] A. H. Khan, X. Cao, S. Li, V. N. Katsikis, and L. Liao, “BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer,” IEEE/CAA Journal of Automatica Sinica, vol. 7, no. 2, pp. 461–471, Mar. 2020, doi: 10.1109/jas.2020.1003048.

[78] A. Singh Saud and S. Shakya, “Evaluation of Weight Decay Regularization Techniques for Stock Price Prediction using Gated Recurrent Unit Network,” Nepal Journal of Science and Technology, vol. 19, no. 2, pp. 9–15, Oct. 2021, doi: 10.3126/njst.v20i1.39379.

[79] D. Yi, J. Ahn, and S. Ji, “An Effective Optimization Method for Machine Learning Based on ADAM,” Applied Sciences, vol. 10, no. 3, p. 1073, Feb. 2020, doi: 10.3390/app10031073.

[80] A. P. Piotrowski and J. J. Napiorkowski, “A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling,” Journal of Hydrology, vol. 476, pp. 97–111, Jan. 2013, doi: 10.1016/j.jhydrol.2012.10.019.

[81] E. H. HSSAYNI, “Regularization of Deep Neural Networks with Average Pooling Dropout,” Journal of Advanced Research in Dynamical and Control Systems, vol. 12, no. SP4, pp. 1720–1726, Mar. 2020, doi: 10.5373/jardcs/v12sp4/20201654.

[82] X. Ying, “An Overview of Overfitting and its Solutions,” Journal of Physics: Conference Series, vol. 1168, p. 022022, Feb. 2019, doi: 10.1088/1742-6596/1168/2/022022.

[83] D. Bacciu and F. Crecchi, “Augmenting Recurrent Neural Networks Resilience by Dropout,” IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 1, pp. 345–351, Jan. 2020, doi:

10.1109/tnnls.2019.2899744.
[84] G. E. Dahl, T. N. Sainath, and G. E. Hinton, “Improving deep neural networks for LVCSR using rectified linear units and dropout,” 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, May 2013, doi: 10.1109/icassp.2013.6639346.
[85] A. Poernomo and D.-K. Kang, “Biased Dropout and Crossmap Dropout: Learning towards effective Dropout regularization in convolutional neural network,” Neural Networks, vol. 104, pp. 60–67, Aug. 2018, doi: 10.1016/j.neunet.2018.03.016.
[86] J. Zeng, M. Zhang, and S.-B. Lin, “Fully corrective gradient boosting with squared hinge: Fast learning rates and early stopping,” Neural Networks, vol. 147, pp. 136–151, Mar. 2022, doi: 10.1016/j.neunet.2021.12.016.
[87] G. Raskutti, M. J. Wainwright, and B. Yu, “Early stopping for non-parametric regression: An optimal data-dependent stopping rule,” 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Sep. 2011, doi: 10.1109/allerton.2011.6120320.
[88] N. Jain, “Optimization of Regularization and Early Stopping to Reduce Overfitting in Recognition of Handwritten Characters,” Lecture Notes on Data Engineering and Communications Technologies, pp. 305–323, 2021, doi: 10.1007/978-981-33-4582-9_24.
[89] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, “Batch normalized recurrent neural networks,” 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar. 2016, doi: 10.1109/icassp.2016.7472159.
[90] A. Ziaee and E. ÇAno, “Batch Layer Normalization A new normalization layer for CNNs and RNNs,” 2022 The 6th International Conference on Advances in Artificial Intelligence, Oct. 2022, doi: 10.1145/3571560.3571566.

[91] N. Keskomon and J. Harnsomburana, "Thai Character-Word Long Short-Term Memory Network Language Models with Dropout and Batch Normalization," International Journal of Machine Learning and Computing, vol. 10, no. 6, pp. 783–788, Dec. 2020, doi: 10.18178/ijmlc.2020.10.6.1006.

[92] V. Sarathi, A. Mujumdar, and D. Naik, "Effect of Batch Normalization and Stacked LSTMs on Video Captioning," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Apr. 2021, doi: 10.1109/iccmc51019.2021.9418036.

[93] L. C. Lee, C.-Y. Liong, and A. A. Jemain, "Validity of the best practice in splitting data for hold-out validation strategy as performed on the ink strokes in the context of forensic science," Microchemical Journal, vol. 139, pp. 125–133, Jun. 2018, doi: 10.1016/j.microc.2018.02.009.

[94] R. J. Meijer and J. J. Goeman, "Efficient approximatek-fold and leave-one-out cross-validation for ridge regression," Biometrical Journal, vol. 55, no. 2, pp. 141–155, Jan. 2013, doi: 10.1002/bimj.201200088.

[95] E. Saccenti and J. Camacho, "On the use of the observation-wisek-fold operation in PCA cross-validation," Journal of Chemometrics, vol. 29, no. 8, pp. 467–478, Jul. 2015, doi: 10.1002/cem.2726.

[96] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," International Journal of Data Mining & Knowledge Management Process, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[97] C. Park, Y. Yang, C. Lee, and H. Lim, "Comparison of the Evaluation Metrics for Neural Grammatical Error Correction With Overcorrection," IEEE Access, vol. 8, pp. 106264–106272, 2020, doi: 10.1109/access.2020.2998149.

[98] M. Ohsaki, P. Wang, K. Matsuda, S. Katagiri, H. Watanabe, and A. Ralescu, “Confusion-Matrix-Based Kernel Logistic Regression for Imbalanced Data Classification,” IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 9, pp. 1806–1819, Sep. 2017, doi: 10.1109/tkde.2017.2682249.

[99] A. González-Ramírez, J. Lopez, D. Torres, and I. Yañez-Vargas, “Analysis of multi-class classification performance metrics for remote sensing imagery imbalanced datasets,” Journal of Quantitative and Statistical Analysis, pp. 11–17, Jun. 2021, doi: 10.35429/jqsa.2021.22.8.11.17.

Appendix A
The Features of Twitter Dataset

has_user_mention	count_of_exclamation_marks	Fofe
count_of_mentions	has_question_mark	has_default_image
count_of_retweets	has_exclamation_mark	has_description
tweet_is_a_retweet	count_of_ellipses	is_verified
tweet_is_a_reply	count_of_special_symbols	length_of_description
Retweeted	used_url_shortner	length_of_screen_name
day_of_week	average_favorite_count	listed_count
length_of_tweet_in_words	average_number_of_hashtags	registration_age
count_of_urls	average_retweet_count	retweet_fraction
length_of_tweet_in_chars	average_tweet_length	status_count
count_of_hashtags	average_urls_mentions_ratio_in_tweets	tweet_time_spacing
count_of_unique_words	count_of_followers	positive_sentiment_score
count_of_unique_chars	count_of_friends	negative_sentiment_score
has_hashtag	favorites_count	has_positive_words
has_url	Fefo	has_negative_words
count_of_question_marks	focus_of_user_on_topic	

Appendix B
The Features of Sentimental LIAR Dataset

ID	party_affiliation	Context	Joy
Statement	barely_true_counts	Sentiment	Disgust
Subject	false_counts	sentiment_score	Sad
Speaker	half_true_counts	sentiment_magnitude	speaker_id
speaker_job	mostly_true_counts	Anger	List
state_info	pants_on_fire_counts	Fear	sentiment_code

الملخص

شــهد العالم في الآونة الأخيرة ما يسـمى بعصـر وسـائل التواصـل الاجتماعي ، حيث يوجد اعتماد كامل على هذه المواقع لمتابعة الأحداث والأنشـــطة. من ناحية أخرى ، تتبع معظم وكالات الأنباء وصـايا مؤسـسـيها أو أصـحابها ، مما يجعل الناس يشـككون في مصـداقية أي مقال اخباري. تكمن المشـــكلة في أن الأخبار المزيفة تركز دائماً على الموضـــوعات الهامة ويتم إصـــدارها في الوقت المناسب ، لذا فهي تنتشر بسرعة وتأخذ صدى واسعاً جداً. على هذا النحو ، ساهم مجتمع البحث في إجراء العديد من الدراســـات للحد من هذه الظاهرة. على الرغم من إجراء العديد من الدراســـات للكشــف عن الأخبار المزيفة باللغة الإنجليزية ، إلا أن تحديد المعلومات المضـــللة باللغة العربية لا يزال غير مطور. تهدف هذه الرسـالة إلى بناء نماذج تعلم آلي متطورة للكشـف عن الأخبار المزيفة باللغتين العربية والإنجليزية. تم تحســين تنبؤات الأخبار المزيفة من خلال ثلاث نماذج مطورة: نموذج التعلم الآلي المطور (DMLM) ، والنموذج العميق الهجين ثنائي الاتجاه القائم على الانتباه (ABHDM) ، ونموذج التعلم العميق الجمعي (EDLM).

النموذج الضبابي FM يتم استخدام لتحديد الميزات المستندة إلى المستخدم ذات الصلة. في البداية ، يتم تقييم هـذه الميزات من خلال طريقتين مقترحتين: متوســط الانخفـاض في مؤشـــر Gini (MDGI) وأهمية السـمة المسـتندة على الاسـقاط (DFI). تم تقييم الميزات المسـتندة إلى المسـتخدم للمرة الثالثة باســـتخدام طريقة الارتباط. ثم يتم تغذية FM بمخرجات هذه الطرق لتحديد الميزات ذات الصـلة. أما بالنسبة لـــ DMLM ، فقد تم تحسين مصنف الغابة العشوائية من خلال التركيز على أفضل الميزات المستندة إلى المستخدم في بناء الأشجار بدلاً من الاختيار العشوائي.

هناك طريقتـان شـــائعتـان لمعالجـة النص همـا الذاكرة ثنـائيـة الاتجـاه طويلـة المدى (Bi-LSTM) والوحدة المتكررة ذات البوابات ثنائية الاتجاه (Bi-GRU). على الرغم من شعبية هذه الشبكات في التعامل مع التسـلســلات التعســفية بكفاءة ، إلا أنها تعامل الميزات المســتخرجة من النص بنفس الأهمية. على هذا النحو ، تم اقتراح ABHDM في هذه الاطروحة لمعالجة هذه المشـــكلة. في هذا النموذج ، يتم تطبيق شـبكة عصـبية تلافيفية (CNN) لتقليل أبعاد مجموعة البيانات. ثم يتم اسـتخدام طبقتين مســـتقلتين Bi-LSTM و Bi-GRU. الفكرة وراء التنفيـذ المتوازي لهـذه الطبقـات هي معالجة النصوص القصيرة والتعسفية في نفس الوقت. يتم تطبيق نموذج الانتباه على مخرجات هذه الطبقات لتحديد الكلمات المفيدة الاكثر فائدة. أخيرًا ، يتم تطبيق منظور متعدد الطبقات (MLP) لتصـــنيف أخبار المقالة على أنها مزيفة أو غير مزيفة. من ناحية أخرى ، اسـتخدمت EDLM كل من Bi-LSTM و Bi-GRU و CNN لتحديد مصـــداقية أو عدم مصـــداقية أي مقال إخباري. تم

بناء هذا النموذج بشكل مشابه لنهج المجموعات المستخدم بشكل شائع في التعلم الآلي. لذلك ، فإن القرار النهائي في تصنيف المقال الإخباري يُتخذ بناءً على آراء جميع هذه الشبكات.

أجريت التجارب على خمس مجموعات بيانات:، Arabic Twitter Dataset ، Sentimental LIAR Dataset ، AraNews Dataset ، AFND ، Fake-or-Real Dataset. تم استخدام الدقة كمقياس لجودة نماذجنا المقترحة ، حيث وصلت دقة DMLM إلى 0.886 و 0.751 لمجموعة بيانات Twitter ومجموعة بيانات Sentimental LAIR على التوالي ، بينما بلغت دقة (ABHDM) 0.8614 و 0.9447 و 0.9944 لمجموعة بيانات AFND و AraNews و Fake-or-Real على التوالي. نتائج التنبؤ الخاصة بـ EDLM اقل بقليل من نتائج ال ABHDM من حيث مقاييس الاداء. علاوة على ذلك ، فإن نتائج نماذجنا المقترحة أفضل مقارنة بالدراسات السابقة التي استخدمت مجموعات البيانات هذه.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم البرمجيات

التنبؤ بالاخبار المزيفة بأستخدام نماذج التعلم الالي المطورة

أطروحة مقدمة إلى

مجلس كلية تكنولوجيا المعلومات جامعة بابل عن استيفاء جزئي لمتطلبات درجة الدكتوراه في تكنولوجيا المعلومات / البرمجيات

من قبل

تحسين علي حسن حميد

بأشراف

أ.د. بان نديم ذنون

1444 هـ

2023 م