Performance Evaluation of Transformer-based NLP Models on Fake News Detection Datasets

by

Raveen Narendra Babu

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

© 2022
Raveen Narendra Babu
Abstract

Fake news has the potential to have catastrophic effects because it is increasingly concerning how it spreads on social media. To identify fake news, various machine learning (ML) techniques have been proposed in recent times. Due to the lack of available research on the performance of various transformer models using datasets that contain data samples from a wide variety of domains, it is essential to increase the research in this field. Hence, this research investigates the performance of various suitable machine learning algorithms implementations on three fake news datasets: LIAR, FNC-1 and Balanced Dataset for Fake News Analysis. Some pre-trained transformer language models, BERT, RoBERTa, ALBERT and DistilBERT, were chosen for this research. The performance of the models utilized in the experiments was consistent across all datasets of varying sizes. The results from the experiments conducted indicated that RoBERTa is the best performing model across all datasets. The results also indicated that DistilBERT trains in half the time required by the other three models. RoBERTa obtained an accuracy of 69% when trained on the LIAR dataset. DistilBERT trained a single epoch within 3.5 minutes, which is significantly faster than what time the other three variants needed to train, 7 minutes. The performance evaluation and the analysis obtained from the results help the research community to advance the investigation and explore insights on fake news detection.
Acknowledgements

I want to express my gratitude and acknowledgement to my supervisor, Prof. Chung-Horng Lung, who helped me to achieve success in this research. I was able to complete my writing phase for my thesis thanks to his directions and counsel. I would like to thank him for providing me relentless support, academically and financially.

I would like to express my thanks to Marzia Zaman and the Research and Development wing of Cistel Technology Inc., for their continued support and providing resources, technical assistance and financial support throughout my research.

I would like to thank Carleton University, for providing me unlimited resources and the infrastructure to complete my research. I appreciate Carleton University for providing me with this opportunity to pursue a research.

Additionally, I want to express my gratitude to my spouse, my family and my friends for their unwavering support and tolerance as I conducted my research and completed my research.
# Contents

Abstract i

Acknowledgements ii

List of Tables vii

List of Figures viii

List of Acronyms xi

1 Introduction 1

1.1 Problem Statement ........................................ 2

1.2 Contributions .............................................. 5

1.3 Organization of Thesis ..................................... 7

2 Related Work 8

2.1 Natural Language Processing (NLP) ........................ 8

2.2 ML Models .................................................. 11

2.2.1 Traditional ML models .................................. 14

2.2.2 Deep Learning based ML models ....................... 14

2.2.3 Transformer based ML models ........................... 15
2.3 Transformer Models ........................................ 16
  2.3.1 BERT ................................................... 19
    2.3.1.1 Pre-training data ............................... 23
    2.3.1.2 Tokenization ................................. 24
  2.3.2 RoBERTa ........................................... 24
    2.3.2.1 Pre-training ................................. 26
    2.3.2.2 Tokenization ................................. 27
  2.3.3 ALBERT ........................................... 27
    2.3.3.1 Pre-training ................................. 30
    2.3.3.2 Tokenization ................................. 30
  2.3.4 DistilBERT ........................................ 30
    2.3.4.1 Pre-training and Tokenization ............... 32
  2.4 Hyperparameters ...................................... 33
    2.4.1 Learning rate .................................... 33
    2.4.2 Optimizer ....................................... 34
    2.4.3 Number of epoch ................................ 34
    2.4.4 Batch ........................................... 35
    2.4.5 Scheduler ....................................... 35
  2.5 Evaluation metrics ................................... 35
    2.5.1 Confusion matrix ................................ 36
    2.5.2 Error rate ....................................... 37
    2.5.3 Accuracy ......................................... 38
    2.5.4 Recall/Sensitivity .............................. 39
    2.5.5 Precision ......................................... 39
    2.5.6 F1 Score .......................................... 40
3 Datasets and Data Preparation

3.1 Fake News Detection Datasets
   3.1.1 LIAR
   3.1.2 FNC-1
   3.1.3 Balanced Dataset for Fake News Analysis

3.2 Data Preparation
   3.2.1 LIAR
   3.2.2 FNC-1
   3.2.3 Balanced Dataset for Fake News Analysis

3.3 Summary

4 Experiments

4.1 Experimental Process

4.2 Environmental Setup

4.3 Python Packages

4.4 Parameters for Transformer Models
   4.4.1 Parameters utilized in the Experiment

4.5 Model selection
   4.5.1 BertForSequenceClassification
   4.5.2 RobertaForSequenceClassification
   4.5.3 AlbertForSequenceClassification
   4.5.4 DistilBertForSequenceClassification

4.6 Summary
5 Analysis of Experimental Results 81

5.1 Experimental Results ................................. 81

5.1.1 BERT ................................................. 82

5.1.2 RoBERTa .............................................. 84

5.1.3 ALBERT .............................................. 86

5.1.4 DistilBERT ........................................... 87

5.1.5 BERT on smaller dataset ............................... 89

5.2 Analysis of Results ...................................... 89

5.3 Summary .................................................. 93

6 Conclusion and Future Work 94

6.1 Conclusion ................................................ 94

6.2 Future Work ............................................. 96

References 98
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Available BERT models (at time of publication)</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>Available RoBERTa models (at time of publication)</td>
<td>26</td>
</tr>
<tr>
<td>2.3</td>
<td>Available ALBERT models</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Available DistilBERT models (at time of publication)</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Data sample of the FNC-1 dataset</td>
<td>49</td>
</tr>
<tr>
<td>3.2</td>
<td>Data split of the LIAR dataset</td>
<td>57</td>
</tr>
<tr>
<td>3.3</td>
<td>Data split of the FNC-1 dataset</td>
<td>60</td>
</tr>
<tr>
<td>3.4</td>
<td>Data split of the Balanced Dataset for Fake News Analysis</td>
<td>63</td>
</tr>
<tr>
<td>4.1</td>
<td>Description of generic parameters for transformer models</td>
<td>74</td>
</tr>
<tr>
<td>4.2</td>
<td>Parameters utilized in specific transformer models</td>
<td>75</td>
</tr>
<tr>
<td>5.1</td>
<td>Training duration (for 1 epoch) of the transformer ML models for the fake news detection datasets (in minutes)</td>
<td>90</td>
</tr>
<tr>
<td>5.2</td>
<td>Evaluation metrics of the transformer ML models on the fake news detection datasets (in %)</td>
<td>90</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Variants of BERT [1] .................................................. 5

2.1 Sample application of Natural Language Processing (NLP) [2] .. 10
2.2 Classification of Machine Learning (ML) models (adapted from [3]) . 12
2.3 Example of encoder-decoder paradigm [?] ............................. 17
2.4 Self-attention concept application[4] .................................. 18
2.5 Pre-training and fine-tuning of BERT [5] ............................... 19
2.6 Sample of pre-training of BERT: Masked Language Model (MLM) and
    Next Sentence Prediction (NSP) [6] .................................... 22
2.7 Transformer model comparison ............................................. 32
2.8 Confusion matrix for binary classification .............................. 37
2.9 Error rate calculation representation [7] .............................. 38
2.10 Accuracy calculation representation [7] ............................... 38
2.12 Precision calculation representation [7] ................................. 39
2.13 Loss function of training set of BERT on the LIAR dataset ......... 41

3.1 Data sample of the LIAR dataset [8] ................................. 44
3.2 Data representation of different classes in the LIAR dataset ......... 45
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Data sample of Emergent dataset [9]</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>Data samples of the FNC-1 dataset</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>Representation of classes in the FNC-1 dataset</td>
<td>50</td>
</tr>
<tr>
<td>3.6</td>
<td>Data sample from the Balanced Dataset for Fake News Analysis [10]</td>
<td>52</td>
</tr>
<tr>
<td>3.7</td>
<td>Representation of classes in the Balanced Dataset for Fake News Analysis</td>
<td>53</td>
</tr>
<tr>
<td>3.8</td>
<td>Data preprocessing of the LIAR dataset</td>
<td>58</td>
</tr>
<tr>
<td>3.9</td>
<td>FNC-1 dataset: A sample output after data preprocessing</td>
<td>60</td>
</tr>
<tr>
<td>3.10</td>
<td>Sample dataset for representing data preparation of the FNC-1 dataset</td>
<td>62</td>
</tr>
<tr>
<td>3.11</td>
<td>Data preprocessing of the Balanced Dataset for Fake News Analysis</td>
<td>63</td>
</tr>
<tr>
<td>4.1</td>
<td>Workflow for training models on fake news detection datasets</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>Model-specific tokenization of a data sample: An excerpt</td>
<td>69</td>
</tr>
<tr>
<td>4.3</td>
<td>Code snippet for encoding the datasets after preprocessing</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>BertForSequenceClassification parameters [11]</td>
<td>77</td>
</tr>
<tr>
<td>4.5</td>
<td>RobertaForSequenceClassification parameters [12]</td>
<td>78</td>
</tr>
<tr>
<td>4.6</td>
<td>AlbertForSequenceClassification parameters [13]</td>
<td>79</td>
</tr>
<tr>
<td>4.7</td>
<td>DistilBertForSequenceClassification parameters [14]</td>
<td>79</td>
</tr>
<tr>
<td>5.1</td>
<td>Performance of BERT on the LIAR dataset</td>
<td>83</td>
</tr>
<tr>
<td>5.2</td>
<td>Performance of BERT on the FNC-1 dataset</td>
<td>83</td>
</tr>
<tr>
<td>5.3</td>
<td>Performance of BERT on the Balanced Dataset for Fake News Analysis</td>
<td>84</td>
</tr>
<tr>
<td>5.4</td>
<td>Performance of RoBERTa on the LIAR dataset</td>
<td>84</td>
</tr>
<tr>
<td>5.5</td>
<td>Performance of RoBERTa on the FNC-1 dataset</td>
<td>85</td>
</tr>
<tr>
<td>5.6</td>
<td>Performance of RoBERTa on the Balanced Dataset for Fake News Analysis</td>
<td>85</td>
</tr>
<tr>
<td>5.7</td>
<td>Performance of ALBERT on the LIAR dataset</td>
<td>86</td>
</tr>
</tbody>
</table>
5.8 Performance of ALBERT on the FNC-1 dataset 87
5.9 Performance of ALBERT on the Balanced Dataset for Fake News Analysis 87
5.10 Performance of DistilBERT on the LIAR dataset 88
5.11 Performance of DistilBERT on the FNC-1 dataset 88
5.12 Performance of DistilBERT on the Balanced Dataset for Fake News Analysis 89
5.13 Performance of BERT on sample of LIAR dataset 89
List of Acronyms

AI . . . . . . . . . . . . . . . . . . . . . Artificial Intelligence

ALBERT . . . . . . . . . . . . . . . . . . . . A Lite BERT

BERT . . . . . . . . . . . . . . . . . . . . . Bidirectional Encoder Representation from Transformers

BiGRNN . . . . . . . . . . . . . . . . . . . . Bi-directional Gated Recurrent Neural Network

DistilBERT . . . . . . . . . . . . . . . . . . Distilled version of BERT

DM . . . . . . . . . . . . . . . . . . . . . . Data Mining

FNC . . . . . . . . . . . . . . . . . . . . . Fake News Challenge

GPU . . . . . . . . . . . . . . . . . . . . . Graphics Processing Unit

GRU . . . . . . . . . . . . . . . . . . . . . Gated Recurring Unit

LSTM . . . . . . . . . . . . . . . . . . . . . . Long Short Term Memory

ML . . . . . . . . . . . . . . . . . . . . . . Machine Learning
MLM ........................ Masked Language Model

NLI .......................... Natural Language Inference

NLP .......................... Natural Language Processing

NSP .......................... Next Sentence Prediction

PCFG .......................... Probabilistic Context-Free Grammars

POS .......................... Part of Speech

QA .......................... Question Answering

RNN .......................... Recurrent Neural Network

RoBERTa ...................... Robustly Optimized BERT Approach

SGD .......................... Stochastic Gradient Descent

SOP .......................... Sentence Order Prediction

SOTA .......................... state-of-the-art

SVM .......................... Support Vector Machine

TF-IDF ........................ Term Frequency-Inverse Document Frequency

TPU .......................... Tensor Processing Unit
Chapter 1

Introduction

Fake news is defined as the dissemination of deliberate misinformation or hoaxes through traditional print, online social media and broadcast news media [15]. Due to the widespread usage of online news, social networking news and other media sources online, fake news has become a serious global issue. Exposure to fake news might lead to cynicism, alienation and attitudes of inefficacy towards particular political candidates [16]. Even violent incidents in the real world that endanger public safety may be related to fake news, such as PizzaGate, according to [17].

Unfortunately, manually moderating or monitoring every news article is impossible with the number of outlets publishing news and the lack of human resources who can moderate it. People do not have the leisure time to check out the authenticity of the news. Hence, it becomes crucial to monitor this issue and eradicate it as much as possible. Fake news can be eradicated by implementing an automatic identification of fake news. Hence, research into fake news detection becomes a priority. The detection of fake news can be accomplished with the help of Machine Learning (ML) models. As a result, the ML research community has increased its efforts in automatic fake
news detection over the last several years.

Studies from the 2016 U.S. elections reveal that social media accounted for an average of 41.8% of all visits to fake news websites during that time. Comparatively, the average traffic share from this type of activity was only 10.1% for legitimate news websites [18]. It is important to note that this data does not reflect the number of fake news headlines or “tweets” that were read without clicking the link [19]. Every American adult is said to have encountered an average of 1 to 3 false news pieces throughout the election season [20].

Automated fake news detection involves determining the veracity of news assertions. The objective of automatic fake news detection is to decrease the time and effort required by humans to identify fake news and aid in its eradication. With the advancement of computer science fields, including ML, Data Mining (DM) and Natural Language Processing (NLP), the task of false news identification has been investigated from a variety of approaches [21, 22, 23]. Because both traditional news and social media have significant social and political effects on every member of society, this is a new but crucial NLP issue [24].

1.1 Problem Statement

There are abundant issues occurring around the world due to fake news propagation. Social media brings unique characteristics and obstacles for fake news detection, rendering typical news media detection algorithms inefficient or irrelevant [25]. Auxiliary information, such as user social engagements on social media, is included to help make a determination because fake news is first deliberately written to mislead readers into believing false information. As a result, it is difficult and nontrivial to detect based
solely on news content. Usage of this supplementary data is difficult in and of itself
due to the large, sparse, chaotic and noisy data that users’ social interactions with
fake news produced. As a result, research into spotting fake news on social media is
currently gaining a great deal of interest.

Due to their different characteristics that could provide insights on the performance
of the ML models, the following benchmark fake news datasets were selected for the
research:

- LIAR [26]: LIAR dataset is a benchmark dataset in the fake news detection
domain due to its data containing a mix of multiple domains: business, political,
sports, etc.

- FNC-1 [27]: Organizers of Fake News Challenge (FNC) developed FNC-1 dataset
[27], a stance detection dataset. This dataset provides an interesting perspective
due to its structure which includes a single true instance and multiple false
instances for a single article body. Hence, this model has been selected to
identify the impact of the structure of the dataset has on the performance of
the model.

- Balanced Dataset for Fake News analysis [10]: This dataset is developed from a
variety of previously developed datasets for fake news detection. The unique
characteristic of this dataset is the balanced nature of the fake and real data
samples. Performance evaluation of the transformer models on this dataset can
provide insights into the impact that the composition of data sample has on the
performance.

The availability of fake news detection modules that utilize transformer models,
a branch of ML models, was negligible. There is a lack of research articles for fake
news detection that use transformer models on a wide variety of fake news detection datasets. Hence, this thesis provides an analysis of performance of transformer models on various benchmark fake news detection datasets. The research evaluates the models based on their accuracy, precision, recall, loss function and the time taken for the model to train on the fake news datasets. The thesis focuses on BERT and its variants as Bidirectional Encoder Representation from Transformers (BERT) was a transformer model primarily developed for text classification [28]. The variants of BERT used for the analysis in this research are:

- BERT [5]
- Robustly Optimized BERT Approach (RoBERTa) [29]
- A Lite BERT (ALBERT) [30]
- Distilled version of BERT (DistilBERT) [31]

BERT is a model that has been developed further into different variants to fine-tune them to specific applications or to improve an aspect to satisfy real-time requirements (refer to Figure 1.1). The thesis emphasizes on these three BERT variants due to their popularity in NLP tasks. These models were also selected for the research due to their ability to have good performance in smaller datasets as well [32]. DistilBERT was developed by distilling the layers and parameters in BERT, reducing the performance of the model, but decreasing the training duration of the model to half the training duration of BERT.
1.2 Contributions

This research is performed to provide a performance evaluation of four different variants of BERT, namely BERT, RoBERTa, ALBERT and DistilBERT, on three fake news detection datasets: LIAR, FNC-1 and Balanced Dataset for Fake News Analysis. The research identifies the best performing model by comparing the accuracy, recall, precision and F1 score metrics (because of imbalanced dataset composition). During this research, every dataset was preprocessed in different ways to provide a data that can be easily used to train the model (refer Section 3.2). Another important aspect that previous research did not concentrate on is the time required for the training of the model. This research calculates the time taken to train the model on each of the three fake news detection datasets to address the lack of research on the training time feature. The experimental results helps to analyse the features that allows the model to improve its performance. The implementation of RoBERTa on
LIAR dataset increases the accuracy of a model on the LIAR dataset by 11%. These features identified can be integrated into other models to develop a hybrid model that has even better features. For example, knowledge distillation feature can be integrated effectively into RoBERTa, the model can perform at high levels and still the training time can be reduced. In addition to the objectives mentioned above, the research concentrates on the impact that the following features have on the performance of the models. There is only little research available on these features and their impact. It is also non-existent on a variety of fake news detection datasets. The features include:

- Size of the training dataset: Experiments have been conducted where only a part of the dataset is utilized for training purposes to check the performance of the model on a smaller data sample for training. During this research 10% of the LIAR dataset was used for training BERT model. The model returned the same accuracy as that of the complete LIAR dataset when evaluated.

- Dependency on the composition of the data samples in the dataset: Experiments have been performed using Balanced Dataset for Fake News Analysis and LIAR dataset, which have very different compositions, i.e., balanced and highly unbalanced samples. The investigation provides insights into the impact of the composition of the data samples in the dataset.

- Effect of structuring of the dataset: Experiments have been implemented on the FNC-1 dataset to obtain a conclusion on the impact of the structuring (refer to Section 1.1) of the dataset on the performance of the model. The experiments comprises of three different datasets: LIAR, FNC-1 and Balanced Dataset for Fake News Analysis, providing a variety of insights which were not found in prior research articles.
1.3 Organization of Thesis

In this section, the structure of this thesis is presented. In Chapter 2, the related work along with backgrounds of the models utilized in the experiments conducted during the research, namely BERT, RoBERTa, ALBERT and DistilBERT, are described along with the concepts. This chapter also introduces the various evaluation metrics and concepts that are deployed in this research to measure the performance of the variants of BERT models. This chapter helps us recognize the lack of research articles for fake news detection that use transformer models on a wide variety of fake news detection datasets. Chapter 3 delves into the datasets and provides a description of them and what they offer to this research. The experimental procedure carried out during the experiments performed for the research is detailed in Chapter 4. The results obtained during the experiments are presented in Chapter 5, along with the detailed analysis obtained from the results of the various experiments conducted. Finally, the thesis is concluded and the future directions that this research could be carried forward are explained in Chapter 6.
Chapter 2

Related Work

Chapter 1 introduced what fake news detection is and what is the objective and motivation for this research. In this chapter, the background and the concepts used in this research to achieve its objective, fake news detection, are elaborately discussed. In Section 2.1, NLP, an important application of ML, that has been utilized in this research, is introduced and explained.

2.1 Natural Language Processing (NLP)

Identification and eradication of fake news is the motivation behind this research. The data, which is fake news, in this task is a textual representation. A field in Computer Science and Artificial Intelligence (AI), known as NLP, is focused on how computers interact with human (natural) languages and how to effectively teach computers to analyse massive volumes of natural language data [33]. Automatic sentiment detection and opinion mining from the text have significantly advanced thanks to NLP algorithms. Due to extensive study in this area, numerous benchmark datasets and methods have been developed. Most NLP approaches before neural networks
mainly concentrated on creating important domain-specific characteristics.

Natural language is seen as any method of everyday human communication, whether it takes the form of writing or speech [34]. Since natural languages are dynamic, defining specific rules for computers is challenging [35]. Programming languages and mathematical notations, regarded as artificial languages, are not included in this category of natural languages.

NLP can be categorized into five main analysis steps, allowing the researchers’ intent to be computationally derived from a textual source. The five stages are:

1. Tokenization-based segmentation: As natural language textual documents typically consists lengthy, convoluted and poorly structured phrases, tokenization is necessary.

2. Lexical analysis: This module converts the word to its base original version (lemmatization).

3. Syntactic analysis: It constitutes finding the relationship of words within the sentences. It is used to analyse if the phrase can be a part of another sentence.

4. Semantic analysis: Semantic analysis works to determine the meaning of individual words, phrases and sentences. As a result, it is frequently used to clarify ambiguities.

5. Pragmatic analysis: Observe pronominal references and the textual coherence of the structure of the adjacent phrases to comprehend a specific sentence.

These five fundamental processes are enough to extract contextualised semantic information from a natural language document, while NLP may also introduce additional stages of analysis, such as emotion recognition [2].
The 2019 oil spill in Brazil reached more than 2,000 kilometers off the coast, affecting marine ecosystems.

Figure 2.1: Sample application of NLP [2]
Figure 2.1 presents a succession of the methods used to conduct segmentation and lexical analysis. Tokenization, removal of punctuation and special characters, elimination of stop words, spelling correction, identification of named entities and stemming or lemmatization are the data cleaning and shaping techniques used in the sequence to perform segmentation and lexical analysis.

In this research, NLP concepts are utilized because of the textual nature of the data. Various NLP techniques like tokenization, stop word removal, etc., are utilized for the data preprocessing. Thus, converting the textual data into a format understandable to the machine is a key part of the research. This is resolved by utilizing existing NLP techniques. In the next section, there is a brief discussion about ML models, especially concentrating on transformer models.

### 2.2 ML Models

The goal of ML, a branch of AI, is to create patterns and regularities. ML offers simpler implementation at low computation costs, as well as quick training, validation, testing and evaluation. It also performs well in comparison to physical models and has comparatively less complexity [36]. Data-driven prediction models using ML are promising tools as they are quicker to develop with minimal inputs [37].

There are crucial aspects of ML algorithms that must be properly considered. First, the systems are only as good as their training, which involves the system learning the intended task based on prior data. This learning is inadequate if the data is insufficient or does not cover all possible variations of the task; as a result, they cannot perform well when put to work. Consequently, it is crucial to use rigorous data enrichment. The second factor is each ML algorithm’s capacity, which may differ for various jobs.
The ability of the trained system to predict scenarios it was not specifically trained for, or if it can forecast outside the parameters of the training dataset, is shown by this challenge, which is also known as a “generalization problem”. Different types of ML models can be observed in Figure 2.2.

![Figure 2.2: Classification of ML models (adapted from [3])](image)

ML could be broadly classified as:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
In this section, the research conducted in the domain of fake news detection using ML models prior to this research is described. There is abundant research available for fake news detection tasks. The existing research includes using traditional and deep learning based ML models like classifier models, regression models Support Vector Machine (SVM), etc.

In this chapter, the research conducted in the domain of fake news detection using ML models prior to this research is described. There is abundant research available for fake news detection tasks. The existing research includes using traditional and deep learning based ML models like classifier models, regression models SVM, etc.

There have been studies that define fake news and its detection. The authors of [18] provide a narrow definition of fake news: fake news is defined as a news story that is purposefully and demonstrably untrue and could lead readers astray. This narrow definition helps the researchers to write the ground truths to their dataset in a universal manner. This allows the researchers to change their research guidelines to create better models. Fake news can be classified into 3 broad categories according to [38]. They are:

- Serious Fabrications
- Large-Scale Hoaxes
- Humorous Fakes

Various researchers have explored the fake news detection through different types of ML models. They are:

- Traditional ML models
- Deep Learning based ML models
2.2.1 Traditional ML models

Various traditional ML algorithms have been developed to detect fake news automatically. The authors in [39] suggest using linguistic-based features for false news identification, including total words, characters per word, frequency of large words, frequencies of phrases, or “n-grams” and bag-of-words techniques. Simple Part of Speech (POS) tagging and content-related n-grams were proved insufficient for the classification job, according to [40]. Instead, they recommended combining deep syntax analysis using Probabilistic Context-Free Grammars (PCFG), suggested by [41], to distinguish between different rule categories for deception detection with 85-91% accuracy. The authors in [42] concentrate on examining the linkages between fake news spreaders and the patterns of fake news dissemination in social networks.

Although bigram Term Frequency-Inverse Document Frequency (TF-IDF) produce effective models for identifying fake news, the author of [43] shows that the PCFG characteristics added little to the models’ performance. Since there may be some correlation between the sentiment of the news story and its kind, many publications have also advocated using sentiment analysis for fraud detection. By evaluating the usefulness of features like part of speech frequency and semantic categories like generalizing terms and positive and negative polarity, the authors in [44] advocated extending the scope of word-level analysis (sentiment analysis).

2.2.2 Deep Learning based ML models

A recent framework based on deep learning for detecting fake news was presented in [45]. They included the structural logic of fake news in the research. Hybrid datasets were
created and utilised for this research during evaluation. Additionally, deep learning models like Long Short Term Memory (LSTM) and a hybrid Bi-directional Gated Recurrent Neural Network (BiGRNN) were used along with ML algorithms like SVM for classification. The accuracy of their work was 82.19%. Gated Recurring Unit (GRU) models and LSTM, which are similar to vanilla Recurrent Neural Network (RNN) models, are suggested in [46] as a classification model for predicting the fake news. Their research made use of the LIAR dataset. In order to compare the accuracy of the results, they produced three trials using Vanilla, GRU and LSTM. The performance of GRU (Accuracy - 21.7%) was superior to that of Vanilla (Accuracy - 21.5%) and LSTM (Accuracy - 21.66%).

To identify misleading news, deep learning models are used in a great deal of research studies, as observed in [47]. In contrast to conventional ML models, the authors of [26] developed a hybrid convolutional neural network model. A detailed investigation of linguistic aspects is conducted, through which encouraging results are obtained using LSTM [48]. To detect fake news, the authors of [22] created a multimodal variational auto-encoder by combining a bi-modal variational auto-encoder with a binary classifier. According to the authors, this end-to-end network uses the multimodal representations derived from the bi-modal variational auto-encoder to categorize whether or not posts are fraudulent.

2.2.3 Transformer based ML models

Transformer models have been extensively used in various text classification tasks in recent years due to their advanced pre-trained knowledge. The authors in [49] use BERT on a semi-supervised pseudo-label dataset and work with a hyperpartisan dataset. The authors of [50] provide a hybrid architecture that uses RNN and BERT
to address the effects of fake news. The authors in [51] propose a transformer-based approach to detect fake COVID-19 news. Experiments were performed on traditional language models and CNN along with transformer models. The dataset is social media posts related to COVID-19 and labels indicating whether the posts are fake or real. They also experimented with transformer-based models and tested different hyperparameters. The highest accuracy is 97.9%, which RoBERTa shows. The research in [51] lacks one important aspect for analysis, a wide variety of benchmark datasets on different domains, which has been included in this research.

2.3 Transformer Models

Transformer models, an unsupervised ML framework, are used in the research because of the lack of comprehensive analysis of performance of transformer models on variety of fake news detection datasets with different compositions and background. Further, the fields of computer vision, computer imaging and NLP have all been revolutionized by transformers. New goods, services and enterprises that depend on transformers to operate, such those created by recent models like GPT-3 and DALL-E, have greatly benefited society [52].

In recent machine translation evaluations, transformers are the state-of-the-art (SOTA) model. The transformer architecture are truly massive, as established in [53]. Larger models cost more to train but deliver better outcomes. Transformers are highly parallelizable, which makes them highly compute-optimal and enables us to train larger datasets. This is a major distinction between transformers and other ML architectures (refer Figure 2.2). The clever usage of attentions [53] is what sets the transformer apart from other ML models. The mapping of a query and a set of
key-value pairs to an output, where the query, keys, values and output are all vectors, is called an attention function. The result is calculated as a weighted sum of the values, with the weights assigned to each value determined by how well the query matches the key in question.

The encoder-decoder paradigm (refer Figure 2.3) is used by the transformer systems. There are multiple identically stacked layers on the encoder side [54]. Each of them is made up of a feed-forward sub-layer and a self-attention sub-layer. The transformer uses a multi-head attention model and the output of this model is fed into a fully connected feed-forward network. The decoder also features a second stack of identical layers. In addition to the two sub-layers utilized in each encoder layer, it also features an encoder-decoder attention sub-layer. In general, the same technique can be used to enhance both the encoder and the decoder because they have similar architectures.

![Figure 2.3: Example of encoder-decoder paradigm](image)

The basic transformer had six layers of encoders and decoders layered on top of one another. Both are given an embedded input with 512-dimensional positional encoding. The input is projected into a 512-dimensional space using multiple attention heads, then projected again into a lower-dimensional subspace with a dimension of 64. A feed-forward network processes the input after the attention function has been applied. The decoder employs a masking method to focus only on tokens up to and including
the present location. In order to build an encoder-decoder mechanism, the decoder also employs the outputs of the preceding decoder layer and the final hidden layer of the encoder stack. Another feed-forward pass is performed after the input has been passed through the two attention layers in the decoder. Before using softmax [55] for the next-token prediction, the entire input is processed through all six layers of the encoder and decoder. After that, a final linear projection is calculated. Each sub-layer uses layer normalization, dropout and residual connection to either speed up or stabilize pre-training.

In order to compute a representation of the same sequence, the attention mechanism known as “self-attention” links various positions of a single sequence. It has been demonstrated to be particularly helpful in abstractive summarization, visual description generation and machine reading [4]. Figure 2.4 provides an illustration of how the self-attention mechanism enables us to discover the relationship between the words being spoken right now and the prior phrase in a sentence.

![Figure 2.4: Self-attention concept application](image)

In the upcoming sections, the models that utilize the transformer architecture, which are utilized in this research are described.
2.3.1 BERT

A transformer-based pre-trained model, called BERT [5] was published by Google AI in October 2018 obtained new state-of-the-art (SOTA) performance on eleven NLP benchmark tasks at the time of publishing. BERT opened the prospect for simultaneously learning left and right word context.

BERT can be used for feature extraction or fine-tuning after learning deep bidirectional representations from the unlabeled text [56]. Furthermore, the model not only performs exceptionally well for tasks that are on the sentence level, like paraphrasing, but also for tasks that are on the token level, such as named entity recognition. In order to help the NLP community, BERT was published in two model sizes in 2018 after it was successfully applied to transfer learning in NLP. The transformer serves as the framework for BERT’s architecture.

The pre-training of the model is conducted on two tasks. They are:

- Masked Language Model (MLM)
- Next Sentence Prediction (NSP)

![Pre-training and fine-tuning of BERT](image)

Figure 2.5: Pre-training and fine-tuning of BERT [5]
Figure 2.5 illustrates that both pre-training and fine-tuning employ the same architectures, with the exception of output layers. Models are initialized with the same pre-trained model parameters for a variety of downstream tasks. Every parameter is adjusted during the fine-tuning process. Every input example now includes the special symbol [CLS], [SEP] and [MASK] token. [CLS] is a special token that indicates the start of a data sample, whereas [SEP] token is used to separate two segments within the same data sample (for example, between question and answer)s. The [MASK] token is utilized to replaced the word that needs to be masked (refer MLM). The following describes the two tasks of model training for BERT, i.e., MLM and NSP.

**MLM:** Since bidirectional conditioning would allow each word to indirectly see itself and the model could predict the target word in a context with multiple layers. The issue of each word indirectly seeing itself is overcome by masking a portion of the input tokens at random. After masking, the model is trained to predict those masked tokens in order to train a deep bidirectional representation. This process is known as a MLM, inspired from *clozing* [57].

Although this enables us to generate a bidirectional pre-trained model, a drawback is that since the [MASK] token does not exist during fine-tuning, a mismatch is caused between pre-training and fine-tuning. This is mitigated by not always replacing “masked” words with the [MASK] token. 15% of the token places are randomly selected by the training data generator for prediction. In the event that the i-th token is selected, either of the following was performed:

- Replace it with the [MASK] token 80% of the time
- Generate a random token 10% of the time
- Leave it intact 10% of the time
Figure 2.6 illustrates the MLM task that is utilized to pre-train the BERT model. For example, the sentence “I play the violin” is considered. After tokenization, it looks like ['I', 'play', 'the', 'violin']. When MLM is applied it the ‘violin’ token is replaced with the [MASK] token. The tokenized sentence after masking is ['I', 'play', 'the', '[MASK]']. The model is now pre-trained with this tokenized set.

**NSP:** Many downstream tasks like Question Answering (QA) and Natural Language Inference (NLI) makes it essential to recognize the ordering of the sentences. Thus, to pre-train the model to fine-tune it later [5] introduced the task of NSP during the pre-training process. Whether or not segment B will come after segment A is predicted in the next-sentence prediction task, thus making it a binary problem. Each training example is a text pair (segment A, segment B). The first segment A is always the starting point. The second segment B, really comes after segment A in 50% of the instances. BERT chooses a segment at random from the entire corpus in the other 50% of occurrences. **IsNext** and **NotNext** are then used to designate the cases, respectively. The two segments are combined and fed to the model as one input sequence. The unique separation token [SEP] identifies them. Figure 2.6 represents how NSP works during the pre-training. The below example is a sentence pair for **isNext** and **NotNext**, respectively.

```
[MASK] MILK [SEP]

ARE FLIGHT LESS BIRDS [SEP]
```
The \texttt{[CLS]} token is used to indicate the start of the sequence and \texttt{[SEP]} token is utilized to separate the segments within a sequence.

![Sample of pre-training of BERT: MLM and NSP](image)

Figure 2.6: Sample of pre-training of BERT: MLM and NSP [6]

The main distinction between BERT and the original transformer version is that BERT only employs stacks of encoders. Six encoder layers are replaced with twelve in the BERT base [5]. In addition, BERT uses a 768-bit embedding size rather than 512-bits. Transformer and BERT both rely on attention processes. BERT solely utilizes self-attention, but the transformer also uses masked self-attention and encoder-decoder attention. BERT implements all key architectural choices, including dropout, residual connection, layer normalization and the attention mechanism. It is simply a stack of the encoder side of the transformer. Table 2.1 tabulates the details of the different versions of BERT available at the time of its publication. Now, there are multiple versions of BERT available publicly, but these two models ($BERT_{base}$ and $BERT_{large}$) remain the main models that the later models are based on.
Table 2.1: Available BERT models (at time of publication)

<table>
<thead>
<tr>
<th>BERT</th>
<th>Layers</th>
<th>Hidden size</th>
<th>Embedding size</th>
<th>Attention heads</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>12</td>
<td>768</td>
<td>768</td>
<td>12</td>
</tr>
<tr>
<td>large</td>
<td>24</td>
<td>1024</td>
<td>1024</td>
<td>16</td>
</tr>
</tbody>
</table>

2.3.1.1 Pre-training data

Both the 800 million-word BooksCorpus and the 2,500 million-word English Wikipedia text passages without lists, tables or headings serve as the basis for BERT’s pre-training. It is emphasized that the value of extracting lengthy continuous sequences from a document-level corpus. A WordPiece model is used to tokenize the data at the beginning [58]. The masking process is then used on each instance in the corpus.

The total length of the input sequences, including all extra special tokens, is set to 512, which means that the maximum number of tokens in a sequence is 512. The authors further decreased computing cost by only using 128 tokens as the sequence length for 90% of the pre-training steps because longer sequences are disproportionately more expensive. The positional embeddings were learned for the remaining 10% of the steps using the “full” sequence length of 512 tokens. The authors do not analyse how much performance is harmed by this. Since the sequence length is set to 512, BERT is pre-trained with a batch size of 256, which yields around 128,000 tokens per batch. This shows that padding the final 384 locations allowed a sequence length of 128 to be used. Adam Optimizer [59] is employed as the optimizer with a learning rate of 1e-4, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. A learning rate warm-up over the first 10,000 steps, an L2 weight decay of 0.01 and a linear decrease of the learning rate are also utilised. All layers employ a dropout probability of 0.1 and GELU (Gaussian Error Linear Units) [60] is used as the activation function. Four Cloud Tensor Processing Unit (TPU) were utilised for BERT base, while sixteen Cloud TPUs were used for
BERT big, resulting in 16 and 64 TPU chips, respectively.

### 2.3.1.2 Tokenization

BERT uses WordPiece embeddings for pre-training and has a vocabulary capacity of about 30,000 tokens. The exact size is determined by the BERT pre-trained version. Instead of looking at complete words, the WordPiece model analyses sub-word units. It is simpler to handle rare words since words are broken into a small number of common sub-words, often known as “wordpieces” [58]. BERT strikes a balance between models that are based on characters and are more flexible and models that look at complete words and are more effective using the WordPiece model. Each sequence begins with a unique `[CLS]` token that is used for classification. This `[CLS]` token is the only one whose hidden state is further processed by the additional fine-tuning layers when performing a classification task for fine-tuning BERT. Along with the token embedding, BERT also picks up a binary segment embedding that only determines which segment each token belongs to and a positional embedding that identifies each token’s exact location within a length of up to 512 tokens.

### 2.3.2 RoBERTa

RoBERTa, Robustly Optimized BERT Approach, was developed by Facebook AI team. It was designed to further optimize the performance of BERT model. The authors of [29] found a few features in the development of BERT that could be further tweaked to improve the performance. They are:

- Static masking
- Redundancy of NSP
Less Pre-training of BERT model

**Static Masking:** The initial BERT implementation only used a static mask because masking was done during data preprocessing. Training data was replicated ten times, resulting in 40 training epochs where each sequence was masked in ten different ways. This duplicate training helps avoid applying the same mask for every training instance in every epoch. As a result, each training session was observed four times while wearing the same mask. This complexity can be removed by using dynamic masking, in which the masking pattern is generated each time a sequence is fed to the model. Dynamic masking becomes more critical when pre-training across a more extensive dataset or with more stages. Since the training set for dynamic masking does not need to be duplicated, this produces results comparable to those of static masking while being more effective.

**Redundancy of NSP:** A number of experiments are performed by the researchers of [29] to identify the importance of NSP in the performance of the BERT model. These experiments were performed due to the researches conducted by [61, 62, 63] that questioned the purpose of NSP. Due to experiments conducted in [29], it was observed that one of the experiments, where NSP is removed performs better than the BERT$_{base}$ in contradiction to results published in [5]. Hence, RoBERTa removed the NSP task during the pre-training.

**Less pre-training of BERT model:** Prior studies in Neural Machine Translation have demonstrated that when the learning rate is adjusted sufficiently, training with very large mini-batches can improve both the speed of optimization and end-task performance [64]. A recent research has demonstrated that BERT can also be trained in big batches [65].

Table 2.2 presents the available versions of RoBERTa model during its time of
publishing. From the table, it can be observed that there is no increase in number of layers or hidden states from that of BERT.

Table 2.2: Available RoBERTa models (at time of publication)

<table>
<thead>
<tr>
<th>RoBERTa</th>
<th>Layers</th>
<th>Hidden size</th>
<th>Embedding size</th>
<th>Attention heads</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>12</td>
<td>768</td>
<td>768</td>
<td>12</td>
</tr>
<tr>
<td>large</td>
<td>24</td>
<td>1024</td>
<td>1024</td>
<td>16</td>
</tr>
</tbody>
</table>

2.3.2.1 Pre-training

Pre-training for RoBERTa differs from that for BERT in that the model is trained for a longer period of time using larger batches of data. RoBERTa is trained on three additional datasets in addition to the BooksCorpus and English Wikipedia data that BERT utilises:

- OpenWebText: Contains web content that was taken from URLs shared on Reddit and at least three upvotes.
- Stories: Incorporates a more story-like context to the model.

Unlike BERT, a dynamic masking method was adopted, hence there was no need to duplicate the dataset. The model was trained using the following parameters:

- Batch size: 8000 (increased from 256 for BERT)
- Learning rate: 6e-4 for RoBERTa\textsubscript{base} and 4e-4 for RoBERTa\textsubscript{large}
- Warm up steps: 24000 warm up steps for RoBERTa\textsubscript{base} and 30000 warm up steps for RoBERTa\textsubscript{large}
- Optimization: Adam Optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.98$

DGX-1 workstations with $8 \times 32$GB Nvidia V100 Graphics Processing Unit (GPU) were used to train RoBERTa.

2.3.2.2 Tokenization

RoBERTa employs an embedding architecture similar to that of BERT, but with a different tokenization approach. GPT-2 introduced the use of byte-pair encoding, which operates at the byte level [66]. A vocabulary with a moderate size of 50,000 can be learnt that can encode a corpus using bytes rather than Unicode characters. The RoBERTa authors chose this byte-level encoding despite the fact that they report slightly inferior performance because of the benefits of a universal encoding technique. Due to the larger embedding matrix and larger vocabulary than for BERT, additional parameters are employed. As a result, there are now 125 million parameters available for RoBERTa basis.

2.3.3 ALBERT

A Lite BERT, ALBERT was developed by Google AI team [30]. Scaling pre-trained model more becomes essential to have a good performance for any model. Unfortunately, continuously scaling a model introduces a lot of problems: computational cost, resource depletion, environmental aspects and model degradation. To overcome all these issues, the authors of [30] had developed ALBERT. This model aims to make scaling of the model easier by reducing the parameters. The parameter are reduced by using two parameter reduction techniques. They are:

- Factorized embedding parameterization: The size of the hidden layers and the
size of the vocabulary embedding are separated by splitting the huge vocabulary embedding matrix into two smaller matrices. Due to this division, it is easier to extend the hidden size without considerably increasing the vocabulary embeddings’ parameter size. Due to this feature, the number of parameters used in ALBERT is only 17% of the parameters that the original BERT model uses.

- Cross-layer parameter sharing: This method stops the parameter from increasing along with the network’s depth. All of the encoder layer parameters in ALBERT are shared.

As usual, many sizes of the ALBERT model architecture have been introduced. ALBERT is also available in xlarge and xxlarge, unlike BERT, which primarily distinguishes between the base and large variants only. Both basic and xxlarge of ALBERT model have an encoder stack with 12 layers; however, xxlarge expands the hidden dimension from 768 to 4,096 and employs 64 attention heads as opposed to the base version’s 12 attention heads. The xlarge and large form comprise 24 encoder layers and hidden sizes of 2048 and 1024, respectively (refer Table 2.3). Only ALBERT variants that employ a larger hidden size than BERT’s 768 are superior to BERT in terms of performance. Only the xlarge and xxlarge versions of ALBERT can perform better than the original BERT model. This performance issue for smaller variants of ALBERT is expected as only parameter sharing will not improve the model’s performance. However, the overall parameters of the model are increased by this increased hidden size. ALBERT can be seen as “lite” compared to BERT when the unique number of parameters is the only factor considered. However, when comparing the performance of various ALBERT model versions to the standard benchmarks, ALBERT only surpasses BERT when a larger hidden size is implemented, increasing
the network’s width.

In addition to the parameter reduction techniques deployed in ALBERT to improve the efficiency of the model, further changes are incorporated into original version of BERT to improve it further. They are:

- Although ALBERT only partially eliminates the NSP loss, it transforms it into an inter-sentence coherence loss. BERT uses two consecutive sentences as positive examples and two randomly chosen sentences as negative examples for the *IsNext* prediction. However, ALBERT flips the order of the positive examples to create the negative ones, arguing that doing so would force the model to learn more nuanced distinctions about discourse-level coherence properties.

- In addition, ALBERT employs a modified version of the Masked Language Modeling task by switching to n-gram masking. The fundamental justification for masking spans instead of individual tokens is that it presents a more challenging problem and improves performance on downstream tasks that demand some level of thinking, like question answering [67]. A maximum of $n = 3$ determines the length of each n-gram mask chosen randomly.

Table 2.3: Available ALBERT models

<table>
<thead>
<tr>
<th>ALBERT</th>
<th>Layers</th>
<th>Hidden size</th>
<th>Embedding size</th>
<th>Attention heads</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>12</td>
<td>768</td>
<td>128</td>
<td>12</td>
</tr>
<tr>
<td>large</td>
<td>24</td>
<td>1024</td>
<td>128</td>
<td>16</td>
</tr>
<tr>
<td>xlarge</td>
<td>24</td>
<td>2048</td>
<td>128</td>
<td>16</td>
</tr>
<tr>
<td>xxlarge</td>
<td>12</td>
<td>4096</td>
<td>128</td>
<td>64</td>
</tr>
</tbody>
</table>
2.3.3.1 Pre-training

ALBERT is trained on the same corpus as BERT, namely BooksCorpus and English Wikipedia. The pre-training was performed using the below parameters:

- Batch size: 4096
- Optimization: LAMB Optimizer [65]
- Warm up steps: 125,000 steps

All models were trained on a Cloud TPU V3 by using 64 to 1024 TPUs for the pre-training.

2.3.3.2 Tokenization

ALBERT requires an input in the format $[\text{CLS}] \ SEGMENT \ A \ [\text{SEP}] \ SEGMENT \ B \ [\text{SEP}]$, similar to that of BERT. The preprocessing methods for pre-training ALBERT are different from that of BERT. SentencePiece tokenization, which ALBERT employs processing of raw text, thus eradicating the problem of language [68]. It specifies the beginning of new words with an underscore.

2.3.4 DistilBERT

DistilBERT, a distilled version of BERT was developed by a team from Hugging Face [31]. This version of BERT was developed with an idea to reduce computational costs, memory requirements and simpler environmental aspects, as most of the models continue to scale up or utilize larger datasets to improve the model performance, neglecting the environmental requirements [69, 70]. DistilBERT aims to reduce the
size of the design while retaining as much of the transformer architecture of BERT as is feasible [71].

- A compression technique called knowledge distillation trains a small model (the student) to mimic the actions of a larger model (the teacher) or an ensemble of models [72]. DistilBERT utilizes knowledge distillation on BERT and tries to mimic BERT performance while decreasing its complexities.

- The general design of DistilBERT is identical to that of BERT; however, [31] has eliminated:
  - Token-type embeddings
  - Pooling layer

- The number of layers are reduced by a scale of 2 (6 encoder layers), while retaining other techniques used in BERT such as a layer normalization and dropout, since they are implemented in a highly efficient manner and important for a stable training.

- DistilBERT is distilled on very large batches using gradient accumulation (up to 4K samples per batch), using dynamic masking and without the next sentence prediction requirement.

- A loss, cosine embedding loss, is introduced over the existing losses present in BERT. This loss has been utilized because of its ability to facilitate an alignment of the hidden state directions of the student and the teacher.

\[ \text{Loss}_{\text{cos}} = -\text{sim}_{\text{cos}}(h_t, h_s) \]
All these measures, when implemented provided a model that was 60% faster, 40% reduced size and still maintaining 97% performance of BERT.

Table 2.4: Available DistilBERT models (at time of publication)

<table>
<thead>
<tr>
<th>DistilBERT</th>
<th>Layers</th>
<th>Hidden size</th>
<th>Embedding size</th>
<th>Attention heads</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>6</td>
<td>768</td>
<td>768</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 2.4 displays the variants of DistilBERT available at the times of its publication.

2.3.4.1 Pre-training and Tokenization

DistilBERT is simply a distilled version of BERT. The token embedding feature to recognize which segment does the token belong to is removed. All other features of pre-training and tokenization remains the same as that of BERT. The pre-training of the model was extremely fast, reducing nearly half of the training time that BERT requires.

<table>
<thead>
<tr>
<th>BERT</th>
<th>RoBERTa</th>
<th>ALBERT</th>
<th>DistilBERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers / Hidden Dimensions / Self-Attention Heads</td>
<td>Base: 12 / 768 / 12&lt;br&gt;Large: 24 / 1024 / 16</td>
<td>Base: 12 / 768 / 12&lt;br&gt;Large: 24 / 1024 / 16</td>
<td>Base: 12 / 768 / 12&lt;br&gt;Large: 24 / 1024 / 16</td>
</tr>
<tr>
<td>Training Time</td>
<td>Base: 8 X V/100 X 12d&lt;br&gt;Large: 280 X V/100 X 1d</td>
<td>1024 X V/100 X 1d (4-5 times of BERT)</td>
<td>1.7 times faster than BERT</td>
</tr>
<tr>
<td>Performance</td>
<td>Outperforming SOTA in Oct 2018</td>
<td>88.5 on GLUE</td>
<td>89.4 on GLUE</td>
</tr>
<tr>
<td>Method</td>
<td>Bidirectional Transformer, MLM &amp; NSP</td>
<td>BERT without NSP, using Dynamic Masking</td>
<td>BERT with reduced parameters &amp; SOP (not NSP)</td>
</tr>
</tbody>
</table>

Figure 2.7: Transformer model comparison

The four variants of BERT that will be used in the experiments for the research have been described in this section and the difference between the four variants are
summarized in Figure 2.7.

## 2.4 Hyperparameters

Hyperparameters are parameters that needs to be fixed before a model can begin its training process. Usually, the performance of the model depends on the configuration of hyperparameters [73]. One configuration of hyperparameters does not satisfy all the cases, similar to one model not fitting all applications. The hyperparameter can be configured based on the:

- Application of the model
- Model utilized
- Dataset setup

In this section, various hyperparameters that are utilized in this research are described.

### 2.4.1 Learning rate

The learning rate is a hyperparameter that determines how much to alter the model each time the model weights are updated in response to the predicted error [74]. It can be difficult to choose the learning rate since a number that is too small could lead to a lengthy training process that may get stuck, but a value that is too large could lead to learning a sub-optimal set of weights too quickly or to an unstable training process.
2.4.2 Optimizer

It is essential to update every epoch weight and the loss function while training a model. An optimizer is an algorithm that can perform the required task. Optimization is challenging as millions of parameters are present in a model [75]. An optimizer is an algorithm or function that can adjust a neural network learning rate and weights. As a result, it helps to increase accuracy and decrease overall loss. There are a few types of optimizers in the ML industry. They are:

- Gradient Descent (GD)
- Stochastic Gradient Descent (SGD)
- Adam
- LAMB
- Adagrad

For this research, both Stochastic Gradient Descent (SGD) and Adam optimizers are considered. Experiments conducted in [76] indicates that Adam optimizer is the most suitable optimizer for a multi-classification problem. Since, this research is based on fake news detection, a multi-classification problem, Adam optimizer is the optimizer utilized in the research.

2.4.3 Number of epoch

The number of epochs, which is a hyperparameter, defines how many times the learning algorithm will run through the full training dataset [77]. One epoch indicates that the internal model parameters have had a chance to be updated for each sample in
the training dataset. There are one or more batches within an epoch. The number of
epochs in our research are limited to two, because of the large size of the dataset and
pre-trained nature of the models.

2.4.4 Batch

The batch size is a hyperparameter that specifies how many samples must be processed
before the internal model parameters are updated. Consider a batch as a for-loop
making predictions while iterating over one or more samples [78]. The predictions are
compared to the anticipated output variables at the batch conclusion and an error is
calculated. The update algorithm is used to correct this inaccuracy, for example, by
moving down the gradient of the error. One or more batches can be formed from a
single iteration of a training dataset.

2.4.5 Scheduler

Scheduler is an added feature over the optimizer in case there is a requirement to
schedule the learning rate in a specific way [79]. Linear scheduler is used in the
experiments for this research.

2.5 Evaluation metrics

Once all these experiments are executed and the models trained, it is imperative to
measure the performance of the model [80]. The performance can be measured in
terms of:

- Error rate
• Accuracy
• Recall
• Precision
• F1-score
• Loss Function

These metrics have been selected due to their usage in previous researches and their importance as a measure of performance of the models. These metrics are calculated with the help of a tabular entity called as Confusion matrix.

### 2.5.1 Confusion matrix

A confusion matrix for a classification problem is a representation of the prediction in a tabular form. The size of the confusion matrix is a \((n \times n)\), where \(n\) is the number of labels involved in the classification problem. Figure 2.8 illustrates a confusion matrix for a binary classification task.

Figure 2.8 provides a graphical representation of the components of the confusion matrix involved in the calculation for error rate of the model. The components of the confusion matrix are:

**True Positive**: True positive samples are those which are actually true and the prediction was true as well.

**True Negative**: True negative samples are those which are actually false and it has been correctly predicted as false.

**False Positive**: False positive samples are those which are actually false, but the prediction is incorrect.
Figure 2.8: Confusion matrix for binary classification

**False Negative**: False negative samples are those which are actually true, but predicted as false.

### 2.5.2 Error rate

Error rate is the ratio of incorrect prediction and total number of samples (refer 2.1). The lowest error rate is 0 (best case scenario) and the highest could be 1 (worst case scenario).

\[
Error \ rate = \frac{FP + FN}{TP + TN + FP + FN} \quad (2.1)
\]
2.5.3 Accuracy

The most popular measurement for the proportion of accurately predicted observations—whether true or false—is accuracy. The following equation may be applied to determine a model’s accuracy (refer 2.2):

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}
\]  \hspace{1cm} (2.2)

In most circumstances, a high accuracy score indicates a good model. However, since a classification model is being trained in this case, a false positive or false negative can have negative repercussions. Likewise, if an article was predicted as fake yet contained factual information, this can undermine trust. As a result, three additional metrics—precision, recall and F1-score, which account for the wrongly classified observation, are employed.
2.5.4 Recall/Sensitivity

The total number of correct classifications of the true class is known as recall. In this research, recall stands for the proportion of articles that were correctly predicted out of all the correctly indicated articles. It can be calculated using the following equation (refer 2.3):

\[
Recall/Sensitivity = \frac{TP}{TP + FN}
\]  

(2.3)

Figure 2.11: Recall/Sensitivity calculation representation [7]

2.5.5 Precision

Precision is the ratio of number of true positives and the total number of true predictions. The equation used for the calculation is (refer 2.4):

\[
Precision = \frac{TP}{TP + FP}
\]  

(2.4)

Figure 2.12: Precision calculation representation [7]
2.5.6 F1 Score

The precision/recall trade-off is represented by the F1-score. It determines the harmonic mean of each pair. As a result, it considers both the false positive and false negative observations. The formula below can be used to determine an F1-score (refer 2.5):

\[
F1_{\text{score}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]  

F1 score is a better indicator for the performance of the model than accuracy in the case of unbalanced dataset. In this research, two of the three datasets are unbalanced, hence the calculation of F1 score is critical. This evaluation metric would play a huge role in identifying which model performs the best. Also with the help of this, it is possible to identify any abnormalities in the model.

2.5.7 Loss Function

The predictions made by the model that is constructed are directly correlated with the loss function [82]. The model will yield great results if the loss function value is low. To enhance the performance of the model, the loss function (or, more precisely, the cost function) must be reduced.

Figure 2.13 presents the loss function of the training of BERT model on the LIAR dataset. The loss function reduces linearly as the epoch increases according to the figure.
2.6 Summary

This chapter has provided the background concepts of all entities included in this research. The summary introduced the concepts of transformer models including the models utilized in this research, hyperparameters (which are capable of altering the models’ efficiency) and the evaluation metrics used in this research. Figure 2.7 summarizes the feature comparisons between the transformer models used in this research.
Chapter 3

Datasets and Data Preparation

In this chapter, the fake news datasets that have been used in the experiments will be discussed along with the methodology used for the data preprocessing.

3.1 Fake News Detection Datasets

This section discusses about the various fake news datasets that will be used in the research. The following datasets have been chosen for this analysis due to their benchmark status in the fake news detection task. They are:

- LIAR [26]
- FNC-1 [27]
- Balanced dataset [10]

These datasets have also been chosen to study the impact in the performance of the model based on the split of REAL-FAKE data samples. The lack of research available about the analysis of performance of transformer models on these benchmark
fake news datasets necessitates this research. Each of these datasets is discussed in the following subsections.

3.1.1 LIAR

The LIAR dataset [26] is a benchmark dataset in the Fake News Detection task. It was developed and published by William Yang in July 2017. It was based out of human-labeled statements that were collected from POLITIFACT.COM’s API [26]. Each of these articles and details have been verified by the editor of the POLITIFACT.COM which provides integrity verification. It contains 12.8K data samples in total. Every data sample in the dataset is labelled as any of the following six items based on the legitimacy of the data.

- pants-fire
- false
- barely-true
- half-true
- mostly-true
- true

Figure 3.1 depicts a single data sample and the components associated with the dataset. From the figure, it can be recognized that each data sample includes information like statement, speaker, context and their corresponding label, which indicates the legitimacy of the data sample.
Figure 3.1: Data sample of the LIAR dataset [8]

The multi-classification task has been converted to a binary-classification for the experimental analysis of the models. The preprocessing procedure for this data is discussed in Section 3.2.1.

The distribution of the various classes of the LIAR dataset can be observed in Figure 3.2. The figure displays both the binary-classification and multi-classification representations. The multi-classification, depicted in Figure 3.2 (a), is converted to the binary-classification (refer Figure 3.2 (b)). The labels ‘true’ and ‘mostly-true’ are grouped to a label ‘1’ (REAL) data sample, whereas all the other classes, namely barely-true, ‘half-true’, ‘false’, ‘pants-fire’ are mapped to label ‘0’ (FAKE). In Figure 3.2 (b), the label ‘1’ refers to the REAL data sample and label ‘0’ refers to the FAKE data sample. With the help of this figure, the ratio between REAL:FAKE is observed as 35:65. This observation leads to the conclusion that dataset is unbalanced. This representation has been obtained by using the seaborn [83] package for visualization purposes.
Figure 3.2: Data representation of different classes in the LIAR dataset
3.1.2 FNC-1

FNC-1 dataset is one of the most comprehensive fake news detection datasets. It was developed as a stance detection dataset. The dataset was created for FNC [27]. The dataset that the organizers of FNC developed, FNC-1 dataset, is based on the Emergent dataset. The origins of the Emergent dataset is explained in this section along with discussion on FNC-1 dataset.

The Emergent dataset [9] was developed for an online journalism project about rumour debunking. There is a website featuring manually validated claims which is available in the work [84] and used as a source for the Emergent dataset. From websites like snopes.com and Twitter accounts notably @Hoaxalizer, rumours regarding a variety of topics, including U.S. and international news as well as technological stories, were collected and utilized for developing this dataset. Journalists first identified the specific assertion from these numerous sources before looking for publications that mentioned it. The journalists then categorized the item as For, Against, or Observing before condensing it into a headline. The authenticity level of the allegation was further assessed as True, False, or Unverified in later phases. There were a total of 300 speculated claims and 2,595 related news stories, with an average of 8.65 articles per claim [56]. The dataset comprises these data which has been manually tagged by journalists with reference to their position and degree of credibility. Figure 3.3 denotes a sample of the Emergent dataset.

Organizers of the FNC randomly matched headlines and article bodies in the Emergent dataset that belonged to various domains in order to develop the FNC-1 dataset. They also matched every article body with its corresponding headline as it was observed in the Emergent dataset. In the FNC-1 dataset, there are 49,972 unique combinations of headline, article body and a label. There are 1,669 distinct
article bodies and 1,648 distinct headlines in total. The organizers included further
instances than observed in the Emergent dataset for the challenge to give the dataset
uniqueness.

The data provided for the FNC include headline, body and stance. The stance
feature are either labelled as agree, disagree, discuss or unrelated based on the
legitimacy of the headline and article. This data is provided as two different files.
They are:

- train_bodies.csv
- train_stances.csv

The organizers of the FNC have provided the test dataset in a similar fashion without
stances: competition_test_bodies.csv and competition_test_stances.csv
[85].

A data sample of both train_bodies.csv and train_stances.csv has
been illustrated in Figure 3.4 (a) and (b), respectively. Figure 3.4 (a) shows that
(a) Dataset sample from FNC-1 (bodies) dataset [85]

<table>
<thead>
<tr>
<th>Headline</th>
<th>Body ID</th>
<th>Stance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soldier shot, Parliament locked down after gunfire erupts at war memorial</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Tourist dubbed &quot;Spider Man&quot; after spider burrows under skin for days</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Luke Somers 'killed in failed rescue attempt in Yemen'</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>BREAKING: Soldier shot at War Memorial in Ottawa</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Comcast Is Threatening To Cut Off Customers Who Use Tor, The Web Browser For Criminals</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Small Meteorite Strikes in Nicaragua's Capital City of Managua</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Breaking: Soldier shot at National War Memorial in Ottawa</td>
<td>0</td>
<td>agree</td>
</tr>
<tr>
<td>Google to buy big chunk of Pacific Shores, iconic Redwood City office park</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Canadian Soldier Shot At Ottawa War Memorial: Report</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>Iraqi social-media rumors claim IS leader slain</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>There has been a shooting at the War Memorial on Parliament Hill. Unconfirmed reports say a soldier was shot.</td>
<td>0</td>
<td>unrelated</td>
</tr>
<tr>
<td>The Pumpkin-Spice Condom Is Just a Figment of Your Own Gross Imagination</td>
<td>0</td>
<td>unrelated</td>
</tr>
</tbody>
</table>

(b) Dataset sample from FNC-1 (stances) dataset [85]

Figure 3.4: Data samples of the FNC-1 dataset
the train_bodies.csv which contains two features: Body ID and articleBody. Similarly, from Figure 3.4 (b), it is observed that the train_stances.csv contains three features: Headline, Body ID and stance. It is observed that for a single Body ID, which is a common feature in both the files (bodies and stances file), there exists one article body, but multiple headlines from the figures.

Table 3.1: Data sample of the FNC-1 dataset

<table>
<thead>
<tr>
<th>Headline: Hundreds of Palestinians flee floods in Gaza as Israel opens dams</th>
<th>Agree (AGR)</th>
<th>Disagree (DSG)</th>
<th>Discuss (DSC)</th>
<th>Unrelated (UNR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hundreds of Palestinians were evacuated from their homes Sunday morning after Israeli authorities opened a number of dams near the border, flooding the Gaza Valley in the wake of a recent severe winter storm. [...]</td>
<td>Israel has rejected allegations by government officials in the Gaza strip that authorities were responsible for released storm waters flooding parts of the besieged area. “The claim is entirely false, and [...]” [...]</td>
<td>Palestinian officials say hundreds of Gazans were forced to evacuate after Israel opened the gates of several dams on the border with the Gaza Strip, and flooded at least 80 households. Israel has denied the claim as “entirely false”. [...]</td>
<td>A Catholic priest from Massachusetts had been dead for 48 minutes before he was miraculously resuscitated. However, it is his description about God that is bound to spark a hot debate about the almighty. [...]</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 illustrates the FNC-1 dataset development. The organizers of the dataset chose a headline and matched it with the correct article body and labelled this pair as Agree (AGR). For the same article body (one data sample), they matched it with different headlines and labelled it accordingly. If the article body disagrees with the headline, they labelled it as Disagree (DSG). If the article body is a random unrelated event to the headline, they labelled it as Unrelated (UNR). Similarly, when the article body was discussing about the headline, they labelled it as Discuss (DSC).
Figure 3.5: Representation of classes in the FNC-1 dataset
The unbalanced nature of the dataset is visualized in Figure 3.5. Figure 3.5 (a) displays the class distribution in the multi-classification scenario. In this image, labels ‘0’, ‘1’, ‘2’ and ‘3’ represent AGR, DSG, DSC and UNR stances, respectively. The number of data samples in the UNR class is very high in the FNC-1 dataset’s class distribution across the four classes. In Figure 3.5 (b), the representation of the data as two classes (Binary-classification) is visualized. In this scenario, labels ‘0’ (AGR), ‘1’ (DSG) and ‘2’ (DSC) are mapped to a label ‘0’ (REAL), whereas label ‘3’ (UNR) is mapped to the label ‘1’ (FAKE). The research is being conducted on the binary-classification task to maintain uniformity between the fake news datasets. From the binary representation, it is ascertained that the REAL:FAKE data sample ratio is 25:75 in the FNC-1 dataset.

### 3.1.3 Balanced Dataset for Fake News Analysis

Balanced dataset for Fake News Analysis has been obtained from kaggle.com. It was developed by Hassan Amin [10]. This dataset was developed due to the lack of balanced fake news detection datasets. The author has collected the data samples from a variety of popular Fake news dataset while maintaining the ratio of FAKE and REAL data samples. This dataset has been widely utilized for regression and classifier models; however, there was little research where it has been used on transformer models. This balanced dataset has been utilized in this research to analyse the impact of data distribution on the performance of the model.

Figure 3.6 depicts a data sample of the dataset. From the figure, it can be realized that the dataset comprises of four features (columns). They are:

- ID
The data samples are either labelled as FAKE or REAL based on the correctness of the title and their corresponding article. If the correct article is matched with their title, the data sample is labelled as REAL. In case of an incorrect match between title and article, the sample is labelled as FAKE.

Figure 3.7 is a representation of the distribution of the dataset between the two classes (REAL and FAKE). From the figure, it can be confirmed that the ratio of REAL:FAKE is 50:50. This indicates that the representation of the classes in this dataset is balanced.
Data Preparation

Data preparation is an essential element in any ML project [86]. It is usually the part where the most time is spent by the researchers to obtain a robust data representation. This data serves as the input to a ML model, hence a robust data can boost the performance of the model. One of the constant issues in data analytics is finding and fixing dirty data and failing to do this can lead to faulty analytics and unreliable judgments [87].

Even though the data preparation is unique for every model or task, it has been observed there are five preliminary preparation methods [86] which are present in most of the projects. They are:

- **Data Cleaning**: Discovering and eliminating faults or inaccuracies in the data.
- **Feature selection**: Determining which input variables are the most important for the task.
- Data Transforms: Changing the variables’ magnitude or distribution.

- Feature Engineering: Generating new variables from the data at hand.

- Dimensionality Reduction: Creating data projections which are compact

The methods to improve the performance of the model, namely data cleaning, feature selection, data transforms, feature engineering and dimensionality reduction, can be realized by utilizing the following concepts and implementing them on the datasets:

- Data split

- Concatenation

- Stop word removal

- Truncation and Padding

- Tokenization

In Sections 3.2.1, 3.2.2 and 3.2.3, the data preprocessing that was performed for the experiment has been discussed. The data preparation tasks have been discussed in general terms below.

**Data split:** The data is usually split into train and test datasets. The data splitting becomes essential as the performance of the model on the valid and test datasets can indicate if the model overfits the data. This observation of overfitting [88] cannot be made if the dataset remains a whole dataset. Overfitting can be observed when the training error is low, but there is a significant increase in the error of the test dataset. This becomes essential for us to evaluate the performance of a model [89]. The model gets trained on the train dataset. When this trained model is applied to the test dataset, the performance of the model can be obtained in an accurate manner.
**Concatenation**: Concatenation is the process of merging two or more features into a single feature [90]. For example, Feature A: `Hello` and Feature B: `World`. When Feature A and B are concatenated another feature is created, say Feature C which has the contents present in both Feature A and B. For the above example, the concatenated Feature C: `HelloWorld`. This is performed to validate the performance of a model on its ability to differentiate between two features when they are concatenated. The concatenation also removes the `[SEP]` token that would be added between the two features when the dataset is tokenized by the transformer model’s tokenizer. This reduces the data space taken by the special tokens, thus increasing the amount of data content in every tokenized data sample. Another method where each sentence can be concatenated was tried by the authors of [29] and the performance was not optimal for usage.

**Stop word removal**: Every model has a limitation of tokens that it can process for the data sample. Any word that is regarded as carrying no information and appearing disproportionately frequently is a stop word. Thus, removal of these stop words becomes an essential part of data preparation. Removing these stop words helps us by reducing the size of the content, hence allowing useful data to be included. There is an abundance of word tokenizers that are capable of the above requirement like NLTK word tokenizer [91], scikit tokenizer [92], etc., Every word tokenizer offers their own set of stop words depending on the application based on which the tokenizer was developed on. This output is served as the input to the next stage - model tokenizer. But, the usage of tokenizers which have their own stop word list could be detrimental to the experiment as certain words in these lists could change the meaning of the sentence, thus affecting the integrity of the data [93]. An example scenario of stop word removal using NLTK tokenizer, which has 179 stop words, is presented below.
When I first met her she was very quiet. She remained quiet during the entire two hour long journey from Stony Brook to New York.

```
input:
text = "When I first met her she was very quiet. She remained quiet during the entire two hour long journey from Stony Brook to New York."
words = [word for word in text.split() if word.lower() not in sw nltk]
new_text = " ".join(words)
print(new_text)
print("Old length: ", len(text))
print("New length: ", len(new_text))
```

Output:
```
first met quiet remained quiet entire two hour long journey Stony Brook New York.
Old length: 129
New length: 82
```

**Truncation and padding:** Truncating becomes a crucial task for preprocessing because all language models have a limit for each instance. For BERT, it is 512 tokens. For each model, there are special tokens like [CLS], [SEP] and [PAD] also included in this limit. Truncation is the process where any data over this 512 tokens gets removed and not used during the further processes in the model. Padding is the process where artificial data is appended to the data sample. The artificially generated data is attached with a special token to allow the model to recognize which is the real and artificial data. This is usually performed when the data sample size is lesser than the size limit of the model. Truncation and padding is performed to increase the performance of the model.

**Tokenization:** Tokenizer algorithms typically make some form of trade-off between having a very flexible character-based tokenization versus a more effective word-based tokenization. Every model has its own tokenizer which can further be selected based on the task (classification, question answering, etc.). For example in BERT model, during the tokenization process, the model introduces special token like [CLS], [SEP] to the dataset. Each feature’s tokens in a single data sample are separated by the [SEP] token and the [CLS] token is used to assess the text pairs. When training, [CLS] is used to determine whether two sentences are sequential, or if in the original text, Feature A comes right after Feature B. The [SEP] token can be used to represent
data samples with multiple features. It can be used by fine-tuning it for additional tasks which require numerous text segments, such as question answering and natural language entailment [96]. These tokenizer algorithm tokenizes the data in a way that the transformer model can best utilize and perform at high levels and provide accurate results.

In the subsequent sections, the data preparation tasks performed for the experiment are explained briefly.

### 3.2.1 LIAR

The effects of each of the data preparation methodologies on the LIAR dataset can be observed in Figure 3.8. In the figure, the base version and the preprocessed version of the dataset is displayed.

For this dataset, the following preprocessing steps were performed on the provided base dataset:

1. Firstly, the dataset is split into three sub-datasets: train, valid and test datasets. The data is split in the ratio of 80:10:10, respectively. Once the data is split and the data is processed in the same way in train, valid and test dataset. The distribution of the data samples (train-valid-test) in its original form is displayed in Table 3.2. The dataset is split in a 80-10-10 (train-valid-test) manner as observed in the table.

<table>
<thead>
<tr>
<th>Table 3.2: Data split of the LIAR dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train dataset size</td>
</tr>
<tr>
<td>Valid dataset size</td>
</tr>
<tr>
<td>Test dataset size</td>
</tr>
</tbody>
</table>
Figure 3.8: Data preprocessing of the LIAR dataset

(a) Before data preprocessing

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TRUE</td>
<td>1972</td>
<td>TRUE</td>
<td>immigration</td>
<td>rick-perry</td>
<td>Governor</td>
<td>Texas</td>
<td>republican</td>
<td>10</td>
<td>30</td>
<td>40</td>
<td>23</td>
<td>18</td>
</tr>
</tbody>
</table>

(b) After data preprocessing

```
<table>
<thead>
<tr>
<th>label</th>
<th>sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>immigration rick-perry Governor Texas republic...</td>
</tr>
<tr>
<td>1</td>
<td>jobs katrina-shankland State representative Wi...</td>
</tr>
<tr>
<td>2</td>
<td>military,veterans,voting-record donald-trump P...</td>
</tr>
<tr>
<td>3</td>
<td>medicare,message-machine-2012,campaign-adverti...</td>
</tr>
<tr>
<td>4</td>
<td>campaign-finance,legal-issues,campaign-adverti...</td>
</tr>
</tbody>
</table>
```

Figure 3.8: Data preprocessing of the LIAR dataset
2. Another feature called ‘label’ is created to convert multi-classification task to a binary-classification task. The ‘label’ feature is populated based on the values of feature in column 1. If the sample belongs to the true, mostly-true class, the value of the data sample in ‘label’ feature is mapped to 1 and it is mapped to 0 for half-true, barely-true, false, pants-fire classes.

3. Unwanted columns like ‘json_id’ (column: 0) and the truth_counts (column: 8:12) are removed. All the empty data points in the dataset are replaced with None value to avoid white space error.

4. Concatenate the features: ‘subject’, ‘speaker’, ‘job title’, ‘state’, ‘party affiliation’, ‘statement’ (column: 2 - 7 and 13) into a feature named ‘label’ to reduce the number of features. Remove the individual features after concatenating the features [90]. This concatenation reduces the number of special tokens that would be introduced by the model’s tokenizer at a later stage in preprocessing.

5. The data samples are truncated and padded due to the requirement of a parameter called max_length in most of the variants of BERT. The data sample is constrained to 256 tokens for the LIAR dataset. This token value is chosen after the dataset is analysed with a box plot and histogram to estimate the size of the data sample, which was 150-200 tokens approximately.

6. The model tokenizers are responsible for the removal of stop and control words. It also adds the special tokens based on the model for the tokenized dataset.
3.2.2 FNC-1

Figure 3.9 depicts a sample outlook of the dataset after all the data preparation tasks necessary for the FNC-1 dataset are implemented. The discussion in this section pertains to the data preprocessing steps applied to the FNC-1 dataset.

<table>
<thead>
<tr>
<th>id</th>
<th>label_bin</th>
<th>text</th>
</tr>
</thead>
</table>
| 0  | 1         | Soldier shot, Parliament locked down after gunfire erupts at war memorial. Small meteorite crashed into wooded area in Nicaragua’s capital of Managua overnight, government said Sunday. Residents reported hearing mysterious boom that left 16-foot deep crater near city’s airport, Associated Press reports. Government spokeswoman Rosario Murillo said committee formed by government to study event determined it was "relatively small "meteorite that "appears to have come off asteroid that was passing close to Earth."

Figure 3.9: FNC-1 dataset: A sample output after data preprocessing

1. Initially, all the four original csv files of the datasets: train_bodies.csv, train_stances.csv, competition_test_bodies.csv, competition_test_stances.csv are obtained from [85]. The train dataset is split as 80:20 (train:valid) sub-datasets. The test dataset is provided with 25,413 samples by the organizers of the FNC themselves. The data split of the datasets is tabulated in Table 3.3

<table>
<thead>
<tr>
<th>Train dataset size</th>
<th>40,350</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid dataset size</td>
<td>9,622</td>
</tr>
<tr>
<td>Test dataset size</td>
<td>25,413</td>
</tr>
</tbody>
</table>

2. Based on the Body ID feature in the files, both the files (competition_train_bodies and competition_train_stances.csv) are merged to create a single file
with three features: Body Id, articleBody and Headline. Similarly, it is repeated for the competition_test_bodies.csv and competition_test_stances.csv to create the test dataset.

3. The features ‘articleBody’ and ‘Headline’ are concatenated in this stage to form a single feature name ‘text’ to reduce the number of special token [SEP].

4. The stances are labelled as 1 (FAKE) for UNR class samples and 0 (REAL) for AGR, DSG, DSC classes samples into a new column ‘label_bin’ (as discussed in Section 3.1.2).

5. NLTK word tokenizer [91] was utilized to remove stop words. It was observed that certain stop words in this tokenizer’s list actually had an impact on the stance. For example: not, nor when caused the stance to be incorrect reducing the data integrity.

6. NLTK word tokenizer with customized stop words was utilized for the stop word removal. The words that were removed included The, the, A, a, An, an. This can be realized when Figure 3.4 and Figure 3.9 are closely observed. For example, data sample with Body ID=0 and the headline="Soldier shot, Parliament locked down after gunfire erupts at war memorial” is represented in Figure 3.10. In the figure, the preprocessed data is represented along with their respective two separate file samples.

7. The data samples are truncated and padded due to the restriction of number of token that the transformer model are able to process. The data sample is constrained to 512 tokens for the FNC-1 dataset. The model is also given information about which tokens are padded and which are not. This is important
The model tokenizer further adds special tokens like [CLS] and [SEP] using their tokenizer to illustrate where the sample is located and to separate between two features, respectively.

### 3.2.3 Balanced Dataset for Fake News Analysis

In this section, the preprocessing steps performed to clean the Balanced Dataset for Fake News Analysis is described. The dataset prior to the preprocessing and after the processing is represented in Figure 3.11.

The data preprocessing for the Balanced Dataset for Fake News Analysis comprises the following steps:
1. Similar to the previously considered datasets, the first step in the data preprocessing is data splitting in this case as well. Due to the lesser number of sample in this dataset, the datasets are split into two sub-datasets:

- Train
- Test

The number of data samples in both train and test sub-datasets, can be visualized in Table 3.4. From the table, it is recognizable that the data is split as 80:20 (train:test). This is very similar to the data split in other datasets utilized during this research. One unique feature of this dataset is, in both train and test datasets, the number of ‘REAL’ and ‘FAKE’ data samples is equal. Every preprocessing step henceforth is performed in both the train and test sub-datasets.

Table 3.4: Data split of the Balanced Dataset for Fake News Analysis

<table>
<thead>
<tr>
<th></th>
<th>Train dataset size</th>
<th>Test dataset size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5,038</td>
<td>1,250</td>
</tr>
</tbody>
</table>
2. Initially the dataset is scoured for empty data points in two columns, namely ‘title’ and ‘text’, as displayed in Figure 3.11 (a). If any data point is observed as empty, it is replaced with *None* value. This is done to eradicate white space error.

3. Two features (columns: ‘title’ and ‘text’) are concatenated into a column named ‘sentence’, as shown in Figure 3.11 (b). Once the ‘sentence’ column is created, ‘title’ and ‘text’ columns are removed from the dataset to remove the redundancy issue.

4. The first feature (column: ‘Unnamed:0’) in the original dataset, as displayed in Figure 3.11 (a), is removed as well as it is not essential for the training of the model.

5. The data samples in the ‘label’ feature are replaced with ‘1’ if it is ‘REAL’ and ‘0’ if it is ‘FAKE’, as can be observed from Figure 3.11 (a) and Figure 3.11 (b), respectively. This is done in order to reduce the number of tokens.

6. Similar to the data preprocessing in the LIAR dataset, model tokenizers are responsible for the removal of stop and control words. It also adds the special tokens based on the model for the tokenized dataset.

### 3.3 Summary

Chapter 4 introduced the various fake news detection datasets that are utilized for the experiments in this research. It also explained how the data is prepared and the various concepts involved in the data preparation including tokenization, stop word removal, data splitting, concatenation, truncation and padding. This chapter also
explains the data preparation performed for the three datasets utilized in the research separately.
Chapter 4

Experiments

In this chapter, the various experiments that were performed during this research is discussed.

The research is aimed at observing the performance of the variants of the BERT: BERT, RoBERTa, DistilBERT and ALBERT on benchmark fake news datasets. The fine-tuning of a pre-trained model to the fake news detection task is the primary task in this research. The datasets used in this research are:

- LIAR dataset [26]
- FNC-1 dataset [85]
- Balanced Dataset for Fake News Analysis [10]

4.1 Experimental Process

In this section, the experimental procedure performed during this research is presented in a detailed manner. The same experimental approach is carried out for all the four models and the datasets with minor changes in DistilBERT implementation. These
changes are necessary due to the absence of two parameters in the DistilBERT model. The parameters absent in the model are:

- `token_id_types`
- `position_ids`

These two elements needed to be eliminated from the code implementation for DistilBERT to execute without any errors.

The experimental procedure followed for each model on a fake news detection dataset is presented in Figure 4.1. The flow chart presents a visual representation of the workflow for the training of a model on a fake news detection dataset. The experimental process is as follows:

1. Model Tokenization: The preprocessed data is fed as an input to the model-specific tokenizer that is mentioned in the previous section. They are:

   - `BERTTokenizer`
   - `RobertaTokenizer`
   - `AlbertTokenizer`
   - `DistilBertTokenizer`

These model-specific tokenizers tokenize the data samples to a form that can be used to train the model. Figure 4.2 presents a data sample before (the first row in Figure 4.2 showing the original input) and after tokenization for all the models. The tokenizers also add unique token IDs in the case of BERT, RoBERTa and ALBERT, as can be viewed in the figure. For DistilBERT, the lack of token IDs can be explained because of its ability to recognize different segments even
Figure 4.1: Workflow for training models on fake news detection datasets
without token IDs (as mentioned in Section 2.3.4.1). The tokenized data for every model differs as the models have their own special tokens and methodology.

Figure 4.2: Model-specific tokenization of a data sample: An excerpt

2. Store token IDs and all tokenized sentences: In the tokenization step of the experiment, the token ID and the tokenized sentences, as shown in Figure 4.2, are stored in a dictionary. Using various parameters in the tokenizers, the maximum length of each data sample is set at 256 tokens, padding is set to the \textit{max\_length} and other parameters: Enabling truncation, returning attention masks and PyTorch tensor conversions. The truncation has been accomplished by the code snippet, as shown in Figure 4.3. This encoded sentences are added to the list \textit{input\_ids}, while the attention masks retrieved are appended to its own \textit{list attention\_masks}. The lists are converted to tensors along with the labels as well for the further processes. The same is implemented to all the model used in the experiments.

Figures 4.3: Code snippet for encoding the datasets after preprocessing

3. Combine all the training inputs (input\_ids, attention\_masks, labels) into a single
tensor dataset. Repeat this process for both the valid and test datasets as well accordingly.

4. Customise the data loader [97] function by selecting the respective tensor dataset and applying the following condition.

- The batch size [77] is set at 32 based on the size of the dataset (LIAR and FNC-1).
- The sampling for the batches is either random: train dataset or sequential: valid and test dataset.

5. Apply the Adam optimizer [98]: The authors in [76] report that Adam optimizer is the most suitable optimizer for a multi-classification problem. Hence, the Adam optimizer is utilized in this research. During the research, the default learning rate of transformer models of $5e^{-5}$ was found to be optimal. Hence, the default setting of learning rate and epsilon value are used throughout the research.

6. Deploy a linear scheduler: The size of the dataset and the batch size along with the limited computational resources available has limited us to initiate the number of epoch as 2. A linear scheduler is deployed in the experiment for the optimization process.

7. Train the model: The data is fed into the model in batches and the time elapsed is retrieved at regular intervals. The loss functions for the datasets are measured in all the datasets: train and valid datasets.

8. Test the model: The test dataset which is encoded is sampled sequentially and loaded into the model. Then the predictions and true label are tracked.
4.2 Environmental Setup

Before starting any research, the environmental setup plays a crucial role in the performance and time requirements for the models. During the initial stages of the research, the experiments were performed on the local machine using the Jupyter Notebook environment. Due to the advantage of cloud resources, which are faster, the research experiments were moved to Google Colab. The availability of a cloud environment and GPU reduced the time required for training significantly. The environment used during the experiments is discussed below.

- IDE: Jupyter Notebook, Google Colab
- GPU: NVIDIA K80/Tesla T4, Python 3 Google Compute Engine Backend
- RAM: 12.68 GB
- Disk Space: 79 GB

In the next section, the packages used during the research are analysed and described in detail.

4.3 Python Packages

During the experiments, different Python packages/libraries are used. The following libraries were installed using the `pip install` command and utilized by calling the `import` command. They are:

- NumPy: NumPy has been utilized in the experiments to analyse and perform mathematical calculation. Using `ndarray`, NumPy allows for a variety of object-oriented approaches, mathematical and logical manipulations with tables [99].
• pandas: This package is utilized for creating and manipulating the data using
dataframes. A pandas series and a one-dimensional NumPy array are extremely
similar, however the series includes additional capability that enables values to
be indexed using labels [100].

• seaborn: The process of understanding data and presenting it graphically or
visually is known as data visualization [101]. During this research, this package
is extensively used to represent the data for bar chart representations.

• Transformers: The transformers package includes all the transformer models
in their pre-trained state for the experiments [102]. For this research, four
transformer models, namely BERT, RoBERTa, ALBERT and DistilBERT are
utilized.

• sklearn: One of the most well-known ML tool kits is scikit-learn [103], a Python
package that combines a variety of cutting-edge ML methods for medium-scale
supervised and unsupervised applications. It is simple for beginners to use in
projects involving ML [104].

• torch: The torch package offers a tensor class that may be used to operate on
multidimensional arrays [105]. The majority of torch’s objects fall under this class.
The tensor supports standard operations like indexing, slicing, transposition and
Basic Linear Algebra Sub-routines (BLAS).

• Plotly express: This package is used for graphical representations in the experi-
ments conducted during this research. During this research, this is employed for
creating a graph depicting the loss function of the model when it is trained on a
dataset.
4.4 Parameters for Transformer Models

In this section, parameters that are utilized in ML transformer models are listed in a tabulated format, briefly describing their purpose. Parameters are used to configure and guide how a particular model should work. The fine tuning of parameters plays a crucial role in the determination of the performance of the model. In Table 4.1, the parameters present in the ForSequenceClassification are represented and described briefly.

Table 4.2 illustrates the parameters that are used by the particular versions of the transformer models used in the experiments. All transformer models used in the research, other than DistilBERT have similar parameters in their fine-tuned model. DistilBERT has removed the requirement for token_type_ids and position_ids.

4.4.1 Parameters utilized in the Experiment

In this section, parameters that have been used in this research are described. All of the parameters are not required for the performance analysis required for the research. The parameters used during the research experiments are:

- *labels parameter* is set at 2 as the experiment deals with a binary classification (REAL, FAKE).
- The *output hidden states* and *output attentions* parameters are set to a boolean False.
- The *input ids* parameter is used to identify the tokens using unique values.
- The *token_type_ids* parameter is used in all models except DistilBERT, as it does not have the parameter in the model as DistilBERT has removed the requirement
Table 4.1: Description of generic parameters for transformer models

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_ids</td>
<td>The indices of input sequence tokens in the vocabulary.</td>
</tr>
<tr>
<td>attention_mask</td>
<td>This parameter is used to indicate if the token is padded or not. It is ‘1’ for non-padded token and ‘0’ for padded tokens.</td>
</tr>
<tr>
<td>token_type_ids</td>
<td>Token indices for the first and second halves of the inputs should be segmented. Indices are chosen as 0 or 1 based on the halves: Sentence A token is represented by 0, while sentence B token is represented by 1.</td>
</tr>
<tr>
<td>position_ids</td>
<td>Position embedding of each input sequence token’s identifiers in the position space. A value between [0, config.max position embeddings - 1] was chosen.</td>
</tr>
<tr>
<td>inputs_embeds</td>
<td>This is used to explicitly pass an embedded representation rather than input_ids. This is helpful if a greater control over the process is required, than the model’s internal embedding lookup matrix uses to transform input ids indices into related vectors.</td>
</tr>
<tr>
<td>head_mask</td>
<td>A mask to disable specific self-attention module heads. Mask values chosen between [0, 1]. If the head is not concealed, it is indicated by 1, whereas if it is covered, by 0.</td>
</tr>
<tr>
<td>output_attentions</td>
<td>Defines whether or not to return the attentions tensors for each attention layer.</td>
</tr>
<tr>
<td>output_hidden_states</td>
<td>Defines whether or not to return all layers’ hidden states.</td>
</tr>
<tr>
<td>return_dict</td>
<td>Indicates whether or not to return a model output.</td>
</tr>
<tr>
<td>labels</td>
<td>Indicates whether to return a model output. Labels used for the classification/regression loss calculation for sequences. The proper format for indices is [0,..., config.num labels - 1]. A regression loss (Mean-Square loss) is computed if config.num labels is equal to 1 and a classification loss is computed if config.num labels is greater than 1 (Cross-Entropy)</td>
</tr>
</tbody>
</table>
Table 4.2: Parameters utilized in specific transformer models

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters in the model</th>
</tr>
</thead>
<tbody>
<tr>
<td>BertForSequenceClassification</td>
<td>input_ids, attention_mask, token_type_ids, position_ids, head_mask, output_attentions, output_hidden_states, return_dict, labels</td>
</tr>
<tr>
<td>RobertaForSequenceClassification</td>
<td>input_ids, attention_mask, token_type_ids, position_ids, input_embeds, head_mask, output_attentions, output_hidden_states, return_dict, labels</td>
</tr>
<tr>
<td>AlbertForSequenceClassification</td>
<td>input_ids, attention_mask, token_type_ids, position_ids, input_embeds, head_mask, output_attentions, output_hidden_states, return_dict, labels</td>
</tr>
<tr>
<td>DistilBertForSequenceClassification</td>
<td>input_ids, attention_mask, input_embeds, head_mask, output_attentions, output_hidden_states, return_dict, labels</td>
</tr>
</tbody>
</table>
for the model to recognize which segment does the token belong to.

- The \textit{attention_mask} parameter is utilized as it becomes necessary for the models to recognize which of the tokens are padded and hence can avoid those tokens during training process.

4.5 Model selection

The fine-tuning task is a classification problem (fake news detection). Hence, the transformer model class for sequence classification is the optimum solution. In this section, the variants of the model classes selected for the experiments are discussed. They are:

- BertForSequenceClassification [11]
- RobertaForSequenceClassification [12]
- AlbertForSequenceClassification [13]
- DistilBertForSequenceClassification [14]

Transformer models like BERT have different variants: bert-base-uncased, bert-base cased, bert-large, etc. It is essential to choose which variant of the transformer models should be utilized for the research. After considering various technical constraints, namely RAM required, GPU performance, etc., the research was decided to be performed using the base models for all the four variants mentioned above. Hence, the following four models were chosen for this research:

- bert-base-uncased [106]
In the subsequent sections, the models chosen for the experiments: BertForSequenceClassification, RobertaForSequenceClassification, AlbertForSequenceClassification and DistilBertForSequenceClassification, are briefly described.

### 4.5.1 BertForSequenceClassification

BertForSequenceClassification is developed by adding a linear layer (sequence classification and regression) over the output pooled layer. From `PreTrainedModel`, this variant of the model descended. This model can be invoked by `BertForSequenceClassification`.

```python
( input_ids: typing.Optional[torch.Tensor] = None, attention_mask:
  = None, inputs_embeds: typing.Optional[torch.Tensor] = None, labels:
  typing.Optional[torch.Tensor] = None, output_attentions: typing.Optional[bool] = None,
) -> transformers.modeling_outputs.SequenceClassifierOutput or tuple(torch.FloatTensor)
```

Figure 4.4: BertForSequenceClassification parameters [11]

The parameters of this model class has been displayed in Figure 4.4. In this experiment, `BertTokenizer` is the tokenizer utilized by the model for converting the dataset to data suitable for training the BERT model.

### 4.5.2 RobertaForSequenceClassification

RobertaForSequenceClassification is developed by implementing a sequence classification/regression layer over the pooled output of the RoBERTa base model [12]. It
inherits characteristics from \textit{PreTrainedModel} \cite{110}. This model can be invoked by \texttt{class transformers.RobertaForSequenceClassification}. The following Figure 4.5 displays the parameters that belong to RobertaForSequenceClassification model. From the figure, it is observed that the parameters of RoBERTa are identical to that of BERT, as shown in Figure 4.4 and Figure 4.5.

![Figure 4.5: RobertaForSequenceClassification parameters \cite{12}](image)

In this experiment, RobertaTokenizer is the tokenizer utilized by the model for converting the dataset to data suitable for training the RoBERTa model.

### 4.5.3 AlbertForSequenceClassification

This is a model that can be invoked by \texttt{class transformers.AlbertForSequenceClassification}. It is developed by adding a linear layer (regression/sequence classification) over the pooled output layer \cite{13}. The parameters of the Albert for sequence classification task is represented in Figure 4.6. The parameters present in Figure 4.6 is identical to that of parameters observed in Figure 4.5.

In this experiment, AlbertTokenizer is the tokenizer utilized by the model for converting the dataset to data suitable for training the ALBERT model.
4.5.4 DistilBertForSequenceClassification

Similar to the previous models in the experiment, this special variation of DistilBERT is developed by adding a linear layer to the pooled output layer [14]. The parameters for this DistilBERT can be obtained from Figure 4.7. As can be observed from the figure, there are only eight parameters in the model. This is because the token_type_ids and position_ids parameters do not exist in the DistilBERT model. There is no need to indicate which segment the token belongs to and the segments can be separated using the [SEP] special token alone.

In this experiment, DistilBertTokenizer is the tokenizer utilized by the model for converting the dataset to data suitable for training the DistilBERT model.

4.6 Summary

In this chapter, various concepts like the environmental setup, python packages and parameters were discussed. It explains the use of each parameter in a tabular format.
in Table 4.2. This chapter summarizes about the parameters that are unique to a particular transformer model (refer Table 4.2). Further, the model selection process was explained and each model briefed. The chapter is concluded with the experimental procedure followed during the experiment conducted for this research. The workflow of the experiment is presented as a visual representation to highlight the concepts utilized for the research.
Chapter 5

Analysis of Experimental Results

This chapter discusses about the various results obtained through the experiments performed during this research. This chapter will provide a comprehensive analysis of the performance of the models: BERT, RoBERTa, ALBERT and DistilBERT, on different fake news detection datasets, namely LIAR, FNC-1 and Balanced Dataset for Fake News Analysis. The differences in each of these datasets leads to providing a better analysis of the impact that the dataset has on the performance of the model. In Section 5.1, the performance of the transformer models, utilized in this research, on the various fake news detection datasets are presented in detail. Section 5.2 provides an analysis of these results obtained from the experiments.

5.1 Experimental Results

In this section, the quantitative results obtained during the various experiments are presented in a pictorial manner for better visualization. The performance of the models when implemented on these datasets can provide crucial insights about the performance of the models. It is possible to recognize the areas where each model
could be improved based on the results obtained. Further sections presents the quantitative results (accuracy, F1 score, recall and confusion matrix) obtained during the experiments.

5.1.1 BERT

The BERT model is the base model for the other three transformer models utilized in this research. Thus, the performance of BERT is a crucial element for analysis purposes. In this section, the results of the performance of the BERT model, when implemented on the fake news detection datasets, are presented.

Figure 5.1 demonstrates the results obtained when BERT is implemented on the LIAR dataset. The LIAR dataset is one of the benchmark datasets in the domain of fake news detection. The number of epochs is limited to 2 because of there was no significant decrease in the training loss graph after 2 epochs and the computational constraints. Figure 5.1 (a) displays the confusion matrix for the BERT model trained on the LIAR dataset. The confusion matrix (in this case) is a representation of true negative, false positive, false negative and true positive (from left to right in Figure 5.1 (a). Figure 5.1 (b) displays all the evaluation metrics that have been obtained for the model, namely accuracy, recall, precision and f1-score. The accuracy of the BERT model trained on the LIAR dataset is found to be 68%. These results are analysed in Section 5.2. Another interesting result for the LIAR dataset performance is the F1 score on the ‘REAL’ (label:’1’) and ‘FAKE’ (label:’0’) data samples. The ‘FAKE’ samples have a high F1 score of 76%, whereas the ‘REAL’ samples have a significantly lower F1 score of 50%. The figure also has a column called support, that displays the number of data samples present in the category.

In Figure 5.2, the performance of BERT model on the FNC-1 dataset is portrayed.
The huge jump in the accuracy in BERT trained and tested on the FNC-1 dataset over BERT trained and tested on the LIAR dataset is one of the critical analysis provided in Section 5.2. As observed from the Figure 5.2 (a), the ratio of number of correct predictions to the total number of predictions are higher than observed in Figure 5.1 (a) (LIAR dataset experiment). The accuracy of this model is observed from Figure 5.2 (b) as high as 98%. The time taken for the FNC-1 dataset is nearly one hour (for 2 epochs), which is very long compared to that of the LIAR dataset, which completes within 14 minutes. This could be accredited to the larger dataset size of the FNC-1 dataset.

In Figure 5.3, the performance evaluation for the BERT model over the Balanced Dataset for Fake News Analysis is observed. Owing to the balanced nature of the
data samples, this results proves to be a crucial element for analysis. The results displayed in the Figure 5.3 (a) indicates that in a total of 1200 test samples, only 40 data samples were predicted incorrectly. This amount of correct predictions has resulted in a high accuracy of 97% represented in Figure 5.3 (b) along with other evaluation metrics.

5.1.2 RoBERTa

RoBERTa is a robust version of BERT. As discussed previously in Chapter 2, it has been established that RoBERTa is theoretically the best of the variants of BERT in this research. The results obtained by RoBERTa by training and testing on the various fake news detection datasets are presented in this section.
Figure 5.4 depicts the evaluation metrics and the confusion matrix of the RoBERTa model over the LIAR dataset. One of the key results to be noted in this figure is the F1-score metric in Figure 5.4 (b). Similar to BERT’s performance on the LIAR dataset, a similar discrepancy is found in the F1 score of ‘FAKE’ (F1 score: 77%) and ‘REAL’ (F1 score: 53%) data samples, where ‘FAKE’ data sample outperforms ‘REAL’ data sample by a significant margin. The training of the RoBERTa model on the LIAR dataset was completed in 14 minutes (for 2 epochs). Looking at the confusion matrix in Figure 5.4 (a), it can be observed that 398 (170 + 128) data samples of the total 1267 test data samples were predicted incorrectly.

Similarly, Figure 5.5 and Figure 5.6 presents the results obtained for the performance of RoBERTa model on the FNC-1 dataset and the Balanced Dataset for Fake News Analysis, respectively. The accuracy results are nearly 100%, as can be observed.
in Figure 5.5 (b) and Figure 5.6 (b). The number of false predictions for RoBERTa on FNC-1 dataset is 335 (211 + 124) out of the total 25413 data samples, indicating the robustness of the model RoBERTa. The time taken to train this model is in the same range as that of BERT model. Further analysis of the performance is explained in Section 5.2.

5.1.3 ALBERT

ALBERT is an acronym for A Lite BERT. The various features of this model discussed in Chapter 2 has informed the nature of ALBERT. A lite version of BERT, that performs nearly at the same level as that of BERT, while decreasing the number of hidden layers in the model is a significant forward step in the ML domain.

The observation of ALBERT follows in the same directions as the previous two models, discussed in Sections 6.1.1 and 6.1.2, in terms of their performance on the LIAR dataset. During the experiment on the LIAR dataset, it is observed that the model has an accuracy of 68% when implemented on the testing dataset (refer Figure 5.7). The same anomaly of the ‘FAKE’ label data samples having a superior performance (as indicated by the F1 score of 77% \(\text{(FAKE)}\) against 44% \(\text{(REAL)}\)) over the ‘REAL’ label data samples continues in ALBERT model as well.
Figures 5.8 and 5.9 represent the performance and evaluation of the model on the FNC-1 dataset and the Balanced Dataset for Fake News Analysis, respectively. The accuracy of the model can be seen as 98% and 97% for the FNC-1 dataset and the Balanced Dataset for Fake News Analysis, respectively. The confusion matrix represent in Figure 5.8 (a) depicts that only 550 (230 + 520) data samples of the total 30,533 test data samples are predicted incorrectly for the implementation of ALBERT on the test dataset of the FNC-1 dataset.

5.1.4 DistilBERT

DistilBERT is a distilled version of BERT, as explained in Chapter 2. The performance of the model on the datasets utilized in this research has been presented in Figures
Figure 5.10: Performance of DistilBERT on the LIAR dataset

(a) Confusion Matrix
(b) Evaluation metrics

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.83</td>
<td>0.72</td>
<td>0.77</td>
<td>941</td>
</tr>
<tr>
<td>1</td>
<td>0.42</td>
<td>0.58</td>
<td>0.49</td>
<td>326</td>
</tr>
</tbody>
</table>

|        | macro avg |       |          |         |
|        | accuracy   | 0.69  | 1267     |

|        | weighted avg |   |          |         |
|        | precision    | 0.63 | 1267     |
|        | recall       | 0.65 | 1267     |
|        | f1-score     | 0.63 | 1267     |
|        | support      | 1267 |         |

Figure 5.11: Performance of DistilBERT on the FNC-1 dataset

(a) Confusion Matrix
(b) Evaluation metrics

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.97</td>
<td>0.96</td>
<td>0.96</td>
<td>7876</td>
</tr>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>18337</td>
</tr>
</tbody>
</table>

|        | macro avg |       |          |         |
|        | accuracy   | 0.98  | 25413    |

|        | weighted avg |   |          |         |
|        | precision    | 0.98 | 25413    |
|        | recall       | 0.97 | 25413    |
|        | f1-score     | 0.98 | 25413    |
|        | support      | 25413|         |

5.10, 5.11 and 5.12.

The performance of this model is slightly lower than that of the other three transformer models. But, one significant result, the training time, has been observed to be halved. This is one of the critical results that will be discussed in Section 5.2, because of its significance and impact on real-time models. DistilBERT too is unable to overcome the anomaly of ‘FAKE’ news data sample superiority in the LIAR dataset. This is discussed further in the next section.

In the next section, the results obtained are analysed and patterns are drawn to compare the performance of the model. The reasons for this difference in performance are analysed and presented to provide a detailed analysis of the performance of the model on these fake news detection datasets.
5.1.5 BERT on smaller dataset

Figure 5.13 presents the confusion matrix and evaluation metrics. The BERT model was trained on the testing dataset of LIAR dataset to verify the performance of the transformer models on a smaller dataset. The results observed were comparable to what was observed when BERT was fine-tuned on the complete LIAR dataset.

5.2 Analysis of Results

In this section, the results obtained from the experiments during the research are analysed. Important results, namely training duration for training each epoch and the accuracy of the model, are presented in a tabular model in Table 5.1 and Table
5.2, respectively.

Table 5.1: Training duration (for 1 epoch) of the transformer ML models for the fake news detection datasets (in minutes)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Transformer Models</th>
<th>BERT</th>
<th>RoBERTa</th>
<th>ALBERT</th>
<th>DistilBERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIAR</td>
<td></td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>3.5</td>
</tr>
<tr>
<td>FNC-1</td>
<td></td>
<td>30</td>
<td>29</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>Balanced Dataset for Fake News Analysis</td>
<td></td>
<td>4</td>
<td>3.5</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.2: Evaluation metrics of the transformer ML models on the fake news detection datasets (in %)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>LIAR</th>
<th>FNC-1</th>
<th>Balanced Dataset for Fake News Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>BERT</td>
<td>68</td>
<td>63</td>
<td>64</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>69</td>
<td>64</td>
<td>65</td>
</tr>
<tr>
<td>ALBERT</td>
<td>68</td>
<td>60</td>
<td>64</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>69</td>
<td>62</td>
<td>65</td>
</tr>
</tbody>
</table>

From Table 5.2, it is observed that RoBERTa is the best performing transformer model on the fake news detection datasets, namely LIAR, FNC-1 and Balanced Dataset for Fake News Analysis. In Table 5.2, various evaluation metrics, namely Accuracy (A), Precision (P), Recall (R) and F1 score (F1) are presented. Table 5.1 presents the duration of the training for the models. It can be observed DistilBERT is the fastest model due to its parameter reduction feature which helps to nearly half the duration for training.
These are some of the important observations collected from analysing the results obtained from the experiments:

- RoBERTa is the best performing model of the models proving its robustness. In RoBERTa, the next sentence prediction (NSP) feature is removed along with the dynamic masking [111] makes it more robust and optimized than BERT and its other variants.

- DistilBERT is the model which trains the fastest in all the fake news detection datasets. Due to the distilled nature of the model because of parameter reduction, hidden layer reduction, etc., this observation is true theoretically as well.

- The performance of the four models on the LIAR dataset has an anomaly. The performance appears to be better when they predict a ‘FAKE’ data sample than the ‘REAL’ data sample. This is observed from the F1 score on the labels ‘0’ (FAKE) and ‘1’ (REAL). This observation causes the performance of the models on the LIAR dataset to decrease by a significant level. This anomaly can be attributed to under-fitting the training for the ‘REAL’ data sample. This anomaly cannot be seen in the other two datasets: FNC-1 and Balanced Dataset for Fake News Analysis, which can be recognized by the uniform F1 scores of the labels.

- The analysis of performances of all the models on the FNC-1 dataset was observed to be high. The models averaged an accuracy of 0.97. This high accuracy is highly owed to the structure of the dataset. The dataset is structured in a way that one article body is matched with different headlines and labelled accordingly. This setup enables the model to train more accurately as it recognized ‘FAKE’ data sample from ‘REAL’ data sample better.
• The analysis of the performance of models on the Balanced Dataset for Fake News Analysis becomes essential for the research to study the impact that the ratio of data samples (REAL:FAKE) has on the performance of the models. This dataset has 50:50 data split with overall 6K data samples. The performance of the models was good with very high accuracy. This observation can be attributed to the ratio of the data sample does play an impact as the model trained on the LIAR dataset, which has a 35:65 ratio, has lower accuracy than the model trained on the Balanced Dataset for Fake News Analysis.

• Transformer model performs in a similar way when the dataset size is reduced. This was observed when the BERT model was trained on the test dataset of LIAR dataset (1000 samples) and performed similar to the results observed in Figure 5.1 (refer Figure 5.13).

• ALBERT is nearly 1.7 times faster than BERT model training [112]. But during the research, it was observed that the model takes relatively the same time as that of BERT in the fake news detection datasets used in this research. This can be attributed to the fact that the number of parameters is reduced, whereas the number of layers remain the same. The version of ALBERT used the textattack/albert-v2-imdb.

• DistilBERT is the model with the least accuracy among the four models in the research. This can be attributed to the nascent stage of development of this model as well as the size of the model. The reduction in the duration of the training period for the model is a significant advantage of this variant of BERT.
5.3 Summary

In this chapter, the results are presented in Section 5.1. Section 5.2 outlines the analysis that was performed from the results retrieved from the performance of the transformer ML models on the three fake news detection datasets: LIAR, FNC-1 and Balanced Dataset for Fake News Analysis.

RoBERTa model is the best performing model based on all the evaluation metrics across all the three fake news detection datasets. The best performance scenario of the entire research is when the RoBERTa model is fine-tuned on the Balanced Dataset for Fake News Analysis with a 100% accuracy observed. From Table 5.1, it is observed that the training duration for DistilBERT is the lowest, nearly half that of all the other transformer ML models used in the research. The chapter also include the analysis that were obtained from the results obtained and the reasons behind the observed results.

In the next chapter, the conclusions derived from the research are presented along with the future directions to this research.
Chapter 6

Conclusion and Future Work

In this chapter, the conclusions derived from the analysis of the results obtained through the experiments during the research are presented. With these conclusions observed, some future directions this research could be carried through are presented in Section 6.2. In Section 6.1, the conclusion of the research is presented.

6.1 Conclusion

From this research perspective, a textual item that is demonstrably incorrect and disseminated with malice is referred to as fake news. It was demonstrated how confirmation bias, the echo chamber effect and basic sociological demands make people usually vulnerable to fake news. A prospect theory perspective explains information ecosystems, where publishers and consumers aim to maximize their utilities while making decisions. Publishers who prioritize short-term audience growth at the expense of consumers whose value is driven more by psychological and sociological factors than by the demand for unbiased information are particularly susceptible to the growth of fake news.
The thesis analyzes the performance of transformer models on three fake news detection datasets. The datasets selected for the research for analysis purposes include LIAR, FNC-1 and Balanced Dataset for Fake News Analysis. The choices of datasets and models was confirmed after rigorous literature review from available research for the fake news detection task.

BERT in 2018 marked the start of the era of largely successful transfer learning in NLP. The transformer architecture was successfully utilised in a bidirectional context for the first time in NLP by BERT. After a literature review was performed, the informed choice for the models was selected from all the transformer models available to the research community. Three variants of BERT along with the base BERT model have been proposed and investigated in this research. The variants included in the research are RoBERTa, ALBERT and DistilBERT. By pre-training the BERT structure excessively and for an extended period of time on more data, RoBERTa further improved BERT’s performance. DistilBERT and ALBERT have questioned usability and were developed with only concentration for lighter frameworks leading to a slight dip in the performance. DistilBERT is developed with a concept called as knowledge distillation to decrease the number of layers. ALBERT can perform better than BERT, but only when the network’s width is increased, creating an upscaled hidden size dimension.

After the implementation and investigation of the models on the fake news detection datasets, it is observed as expected that RoBERTa is the best performing variant of BERT. This can be attributed to the increased time and data that RoBERTa is pre-trained on. The careful choosing of the dataset has helped to gain insights about the impact of the composition of the data sample. There is a huge difference in the performance of models on the LIAR dataset and the Balanced Dataset for Fake News
Analysis. The improved performance can be attributed to the composition of the dataset as the performance on the LIAR dataset was very imbalanced on ‘REAL’ and ‘FAKE’ data samples. The under-fitting of the ‘FAKE’ data sample is the reason identified for this discrepancy of performance on the LIAR dataset. DistilBERT was observed as the model that took the least minute to train on all the three fake news detection datasets. This observation was expected due to the reduction in the size of the model through the concepts discussed in Section 2.3.4. The reduction in duration taken for training for ALBERT did not happen as expected theoretically. Though their performance was nearly the same as that of BERT model (comparatively less), the time required to train ALBERT remained the same as that of BERT. This can be attributed to the reduction is parameter-wise only and not layer-wise in ALBERT. This observation provides a conclusion that the training time depends on number of layers.

6.2 Future Work

In this section, the future directions the research can be enhanced are discussed.

- The parameter reduction of ALBERT can be utilized in RoBERTa to decrease the training model time, but still maintaining the performance of RoBERTa.

- Integrating or creating a hybrid model with the best features of all these models are an area that is being researched. This improvements can be derived from the observations of this research.

- With the advent in graph databases in the NLP research community and its advantages, future research can be performed on a dataset represented in a
graphical manner. The improvement in the retrieval time of the graph databases is a feature that could be used to developed applications with rapid responses.

- The analysed models can be deployed in a real-time environment over the web to create a fake news detection portal with RoBERTa being the model utilized and the model fine-tuned on FNC-1 dataset.

- The conclusion from the LIAR dataset can be further researched and the discrepancy eradicated.

- The concept of Sentence Order Prediction (SOP) which is utilized, instead of NSP, in ALBERT can be researched further. The concept has a lot of potential in the field of NLP to improve the efficiency of a ML model.

- The number of epochs for training can be enhanced to verify if the training loss can be further minimized.

- Due to resource constraints, only the base models of the models were utilized for the research. For this research, since base models of all the four variants are utilized, the results are conclusive. If the research could further be enhanced by utilizing large and xl versions of the models, further insights or discrepancies could be observed.
References


108


