1. Introduction

Fake News is false or misleading information that is presented as news. With advancements in today's technologies, there is abundance of information available across various digital platforms, but there are no proper channels to filter or validate this information. Hence, users often do not have an idea whether the given news article is fake or real. Some studies also suggest that fake news spread faster than real news [1]. Also, fake news is capable of disturbing governments and therefore can negatively impact societies and individuals. There is a lot of on-going research to detect fake news automatically making use of advanced machine learning and artificial intelligence techniques.

This project is based on fake news detection, where we have conducted a comparative study of advanced pre-trained natural language processing models like BERT [2] (Bidirectional Encoder Representations from Transformers) and its optimized version called RoBERTa [3] (Robustly optimized BERT training approach) using LIAR [4] dataset. In simple terms, we are planning to test the performance of these two algorithms in classifying a given news article into fake or real categories.

2. Literature Review

There are various studies that have been done in fake news detection. Most of these studies are based on various CNN and LSTM based architectures. Some researchers [5] developed models based on BERT architecture, where the authors have used some extra data apart from Wikipedia and Books corpus data in the pretraining stage of BERT to improve its detection capabilities. Some other studies [6] proposed models by combining different parallel blocks of single layer CNN with BERT and achieved better results than existing models on real world fake news dataset. Some other studies [7] compared different models based on LSTM, CNN, and BERT on Fake News Corpus, where BERT performed well with 98% test accuracy. All these studies proved that BERT based models are efficient in detecting Fake News.

3. LIAR Dataset

LIAR dataset [4] was published by William Yang in July 2017. It is one of the challenging publicly available datasets for Fake news detection. LIAR dataset is a collection of 12,791 manually labelled short statements from POLITIFACT.COM. Each of these statements are evaluated by editors of politifact.com for its truthfulness. Based on the truthfulness, each of the statements were categorized into six labels ranging from True to Pants on Fire. Dataset basically consists of 14 columns with features like Json Id of the statement, labels, the statement – title and often the statement, subject representing the subjects of the statement, speaker representing the source of the statement, speaker's job, the US state where they are based, the party he/she is affiliated to, the count of truth values for the speaker including the current statement and the context of the statement representing the venue or location of the statement. I have downloaded this dataset from Kaggle [8]. This dataset is pre-divided into train, validation, and test sets with 10240, 1284 and 1267 statements, respectively.

3.1 Data Preprocessing

For this project, I have modified the label column into only 2 categories – Real and Fake. All true and mostly true statements were classified as Real statements and half-true, mostly false, false and pants on fire were classified into Fake category statements. Also, I have dropped unwanted columns like ID and truth counts, since including the truth counts may influence prediction, as we are feeding the algorithm

already that a particular speaker gives true or false statements most of the time. Also, as part of data cleaning, I have replaced the empty data points with 'None'. After this step, I have combined the subject, speaker, job title, state, party affiliation, statement columns into a single column called 'sentence' and used this column for training. I have dropped the individual columns and retained just the label and sentence columns. The structure of the data after all pre-processing steps is shown below in Fig 1.2.

Column(s)	Description
id	The json ID of the statement
label	Truth value of the statement; 6 categories from 'true' to 'pants on fire'
statement	Title of the PolitiFact article, often but not always the actual statement
subject	The subject(s) of the statement
speaker	The source of the statement
speaker_job speaker_us_state speaker_affiliation	The speaker's job title, US state where they're based, and party affiliation, where available
speaker_bt … speaker_pof (5 features)	Total count of truth values for the speaker (truth credit history), excluding 'true' count and including the current statement
context	The context (venue / location of the speech or statement)

Figure 1.1: Dataset Description

	label	sentence
0	0	abortion dwayne-bohac State representative Tex...
1	0	energy,history,job-accomplishments scott-surov...
2	1	foreign-policy barack-obama President Illinois...
3	0	health-care blog-posting nan nan none a news r...
4	0	economy,jobs charlie-crist nan Florida democra...

Figure 1.2 Dataset Structure after Preprocessing

I have also analyzed the lengths of sequences using boxplot and histogram, as both BERT and RoBERTa requires a parameter called max_length, which represents the maximum length of the sequence. The plots for this analysis are included in Appendix (Fig 6.1,6.2). After analyzing these plots, I have observed that most of the sequences are of length 150-250. Hence, I have decided the max_length parameter to be 256. If the length of the sequence is less than 256, tokenizer adds padding elements and if it is greater than 256, tokenizer truncates the sequence to 256 tokens.

4. BERT

Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art model, published by researchers at Google AI. Applying Bidirectional training of transformer to a language model is the main technical innovation of BERT. It is pretrained on Wikipedia and Books Corpus, which is around 3,300 million words and requires task specific tuning. Since, the model is trained on a large text corpus, it helps it in understanding the language deeply and helps to grasp the relationship between words, which makes it a powerful tool for any Natural Language Processing related task. It is pretrained on two different tasks – Masked Language Modelling (MLM) and Next Sentence Prediction (NSP).

Masked Language Modeling (MLM) is useful to train the model understand the relationship between words. This process randomly masks a set of words (using masked token [MASK]) in each sequence and tries to predict those masked words. To prevent the model from focusing too much on a particular position or tokens that are masked, researchers of BERT randomly masked 15% of the words. But since the masked token [MASK] would never appear during fine tuning i.e., in real time, researchers used techniques like

replacing the words with masked token 80% of the time, 10% of the time they are replaced with random words and 10% of the time the words are left unchanged.

Next Sentence Prediction (NSP) helps the model to understand the relationship between sentences. It is a binary classification task, where given a pair of sentences A and B, the model predicts if B is the next sentence of A. The data can also be easily generated by splitting the sentences into pairs. For 50% of the pairs, B is the next sentence of A, which will be labelled as 'IsNext' and for other 50% of the pairs, B will be a random sentence, which will be labelled as 'NotNext'.

The set of sentences in Fig 2 demonstrates the MLM and NSP tasks. In the first pair of sentences "the man went to the store. he bought a gallon of milk", words 'the' and 'of' are replaced by masked token [MASK]. Also, the second sentence is the next sentence of the first one. Hence the label is "IsNext". In the second pair of sentences, "the man went to the store. Penguins are flightless birds", the words, 'went' and 's' are masked. Here, the second sentence "Penguins are flightless birds" is not related to the first sentence. Clearly, the second sentence is not the next sentence of the first one. Hence the label is "NotNext".

```
Input = [CLS] the man went to [MASK] store [SEP]
        he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
        penguin [MASK] are flight ##less birds [SEP]
Label = NotNext
```

Figure 2: MLM and NSP example

4.1 BERT Architecture

Bert has two variants – Base and Large. I have used BERT base model for this project. BERT base has 12 layers of transformer blocks, 12 attention heads and 110 million parameters. BERT is based on transformer architecture. BERT is 'Bi-directional', implying that the model learns from both left and right contexts of the token. For example, consider the two sentences in Fig 3. In the first sentence, the meaning of the word bank is determined by its context (We went to the river) which is to the left of the word. In the second sentence, the meaning of the word bank is determined by its context (make a deposit) which occurs to the right of the sentence. Unidirectional models learn from either left to right or from right to left, which makes the model less efficient in learning these types of examples. In such cases, the model will be able to learn only one of these examples correctly. BERT on the other hand learns from both right and left contexts, which makes it more efficient when compared to other models.

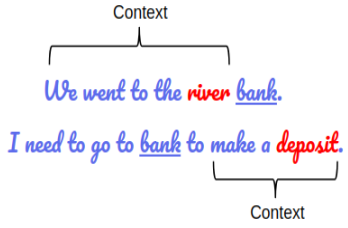

Figure 3: Left and Right Contexts

BERT is based on transformer architecture. Transformer structure basically contains Encoder and Decoder.

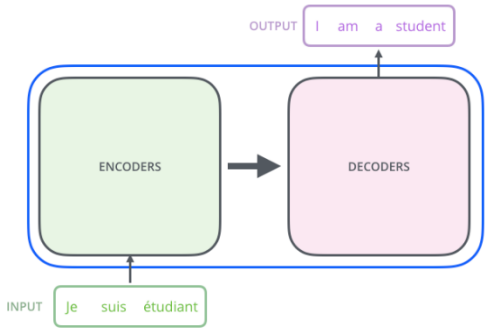

Figure 4: Transformer

Encoder part of the transformer takes the input, processes it, and converts it into an intermediate state, which will be passed as an input to the Decoder. Decoder then decodes the intermediate state into the required output. Apart from this, transformers use something called attention mechanism, which provides weights to each of the input tokens based on their importance in the sentence. These weights are called 'attention weights'. Along with the intermediate state, attention weights are also used by the decoder during decoding the sentence. Usage of attention weights helps the model understand the important words and thereby its context and thus helps in understanding the context of the sentence, which is particularly useful in machine translation tasks to translate the sentences without losing their meaning and context.

The usage of all these techniques makes BERT very efficient and powerful tool that can be used for any NLP related task.

5. RoBERTa

Robustly optimized BERT training approach (RoBERTa) was introduced by researchers at Facebook. This model is an improvised version of BERT, where it is trained using larger datasets, using high computational power. Apart from using Wikipedia and Books Corpus data, the authors of the paper have also used Common Crawl News data and Open Web text data in the pre-training part, which equaled about 160 GB of uncompressed text. In the Masked Language Model task, researchers used the concept of dynamic masking, where masking pattern is generated every time, a sequence is fed to the model, which yielded better results. For Roberta, researchers removed the Next Sentence Prediction (NSP) task from its objective, which improved the performance of the downstream task. Also, it is trained using large batch sizes and longer sequences. Using all these techniques, researchers have optimized the BERT model, making it more robust for fine tuning it for downstream tasks.

6. Input Preprocessing

Since BERT and RoBERTa are pre-trained models, they accept input only in a certain format, vectors of integers, where each integer value represents a token. This input pre-processing steps are done by the tokenizers. They accept the input sequences and converts them into the required formats as needed by the models. Both BERT and RoBERTa have their own tokenizers. These tokenizers add the special tokens representing the start and end of a sequence, tokenizes i.e., converts a given sequence into tokens. These tokens can be words, sub words or characters. It also assigns a unique ID to each of these tokens and replaces those tokens with the token IDs wherever the token appears in the corpus. Thus, it creates the vector of integers as required by the models.

7. Results

The pretrained models are available in the hugging face's transformers library. I have loaded these pretrained models and fine-tuned them using our dataset by training the model for 2 epochs using a batch size of 32. Since this is a binary classification problem, I have used classification report and confusion matrix to analyze the results. I tried experimenting the model by training for 2, 3 and 5 Epochs for BERT. But I have observed that the value of training loss reduced from 3rd epoch, but validation accuracy declined, which implies model started to overfit beyond 3 epochs. The results of training the model for 5 epochs are shown in Fig 7-Appendix. To avoid the problem of overfitting, I have decided to train the model for just 2 epochs for the models for optimal results. Below are the classification report and confusion matrix for BERT and RoBERTa on the test set, which consists of 1267 statements.

BERT

	precision	recall	f1-score	support
0	0.80	0.72	0.76	904
1	0.44	0.54	0.49	363
accuracy			0.67	1267
macro avg	0.62	0.63	0.62	1267
weighted avg	0.69	0.67	0.68	1267

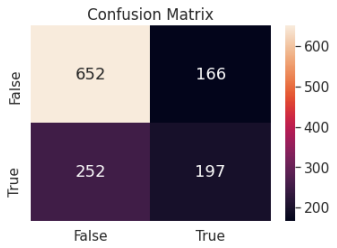

Figure 5.1: BERT Classification Report and Confusion Matrix

RoBERTa

	precision	recall	f1-score	support
0	0.77	0.74	0.76	854
1	0.51	0.55	0.53	413
accuracy			0.68	1267
macro avg	0.64	0.65	0.64	1267
weighted avg	0.69	0.68	0.68	1267

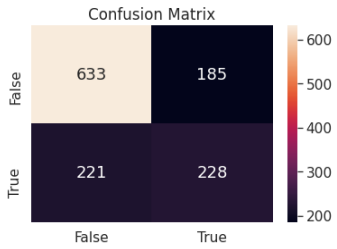

Figure 5.2: RoBERTa Classification Report and Confusion Matrix

From the above results, we can observe that of 1267 statements, BERT predicted 904 statements as Fake and 363 statements as Real. The model performed well in terms of determining the Fake news, but less effective in terms of determining Real news. The overall accuracy achieved is 67% using BERT model. RoBERTa achieved an accuracy of 68% on the test set. We can also observe that of 1267 statements, 854 statements were classified into Fake category and 413 statements were classified into Real Category. This implies that RoBERTa performed well in achieving the trade-off between Fake and Real categories better than the BERT model and achieved slightly better test accuracy than BERT model.

8. Conclusion

Overall, both the models performed better than the results obtained in the LIAR dataset, where the authors achieved an accuracy of 27% on the test set using Hybrid CNNs. RoBERTa performed slightly better on the test dataset with an accuracy of 68% when compared to BERT model which achieved an accuracy of 67%. Also, RoBERTa performed better in achieving the trade-off in classifying between Fake and Real categories.

REFERENCES

[1] https://www.bbc.com/news/technology-43344256

[2] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT*. https://arxiv.org/abs/1810.04805v2

[3] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://arxiv.org/abs/1907.11692

[4] Wang, W.Y. (2017). "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. *ACL*. https://arxiv.org/abs/1705.00648

[5] Jwa, H.; Oh, D.; Park, K.; Kang, J.M.; Lim, H. exBAKE: Automatic Fake News Detection Model Based on Bidirectional Encoder Representations from Transformers (BERT). *Appl. Sci.* 2019, *9*, 4062. https://doi.org/10.3390/app9194062

[6] Kaliyar, R.K., Goswami, A. & Narang, P. FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. Multimed Tools Appl 80, 11765–11788 (2021). https://doi.org/10.1007/s11042-020-10183-2

[7] Rodríguez, Á.I., & Iglesias, L.L. (2019). Fake news detection using Deep Learning. *ArXiv, abs/1910.03496*. arXiv:1910.03496

[8] https://www.kaggle.com/khandalaryan/liar-preprocessed-dataset

[9] https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/#:~:text=%E2%80%9CBERT%20stands%20for%20Bidirectional%20Encoder,both%20left%20and%20right%20context.

[10] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

APPENDIX

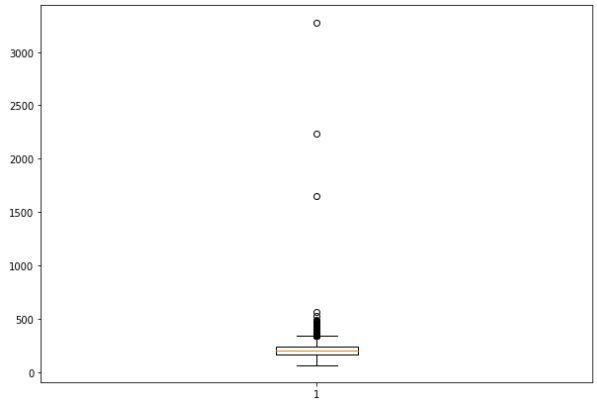

Figure 6.1: Boxplot of sequence lengths

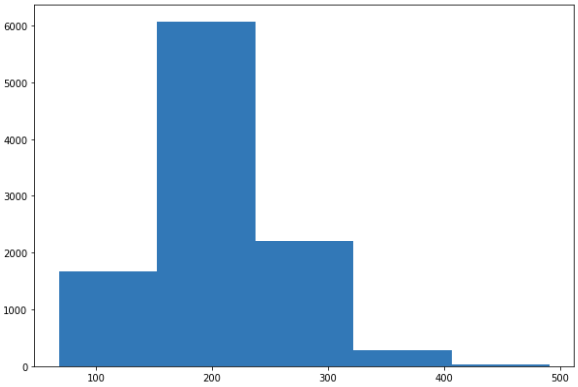

Figure 6.2: Histogram of sequence lengths

```
======== Epoch 1 / 5 ========
Training...
  Batch    40  of    288.    Elapsed: 0:01:36.
  Batch    80  of    288.    Elapsed: 0:03:12.
  Batch   120  of    288.    Elapsed: 0:04:49.
  Batch   160  of    288.    Elapsed: 0:06:25.
  Batch   200  of    288.    Elapsed: 0:08:01.
  Batch   240  of    288.    Elapsed: 0:09:37.
  Batch   280  of    288.    Elapsed: 0:11:13.

  Average training loss: 0.64
  Training epcoh took: 0:11:32

Running Validation...
  Accuracy: 0.69
  Validation took: 0:00:29

======== Epoch 2 / 5 ========
Training...
  Batch    40  of    288.    Elapsed: 0:01:36.
  Batch    80  of    288.    Elapsed: 0:03:12.
  Batch   120  of    288.    Elapsed: 0:04:48.
  Batch   160  of    288.    Elapsed: 0:06:25.
  Batch   200  of    288.    Elapsed: 0:08:01.
  Batch   240  of    288.    Elapsed: 0:09:37.
  Batch   280  of    288.    Elapsed: 0:11:14.

  Average training loss: 0.57
  Training epcoh took: 0:11:33

Running Validation...
  Accuracy: 0.70
  Validation took: 0:00:29

======== Epoch 3 / 5 ========
Training...
  Batch    40  of    288.    Elapsed: 0:01:36.
  Batch    80  of    288.    Elapsed: 0:03:13.
  Batch   120  of    288.    Elapsed: 0:04:49.
  Batch   160  of    288.    Elapsed: 0:06:26.
  Batch   200  of    288.    Elapsed: 0:08:03.
  Batch   240  of    288.    Elapsed: 0:09:39.
  Batch   280  of    288.    Elapsed: 0:11:16.

  Average training loss: 0.39
  Training epcoh took: 0:11:36

Running Validation...
  Accuracy: 0.67
  Validation took: 0:00:30

======== Epoch 4 / 5 ========
Training...
  Batch    40  of    288.    Elapsed: 0:01:37.
  Batch    80  of    288.    Elapsed: 0:03:14.
  Batch   120  of    288.    Elapsed: 0:04:52.
  Batch   160  of    288.    Elapsed: 0:06:30.
  Batch   200  of    288.    Elapsed: 0:08:07.
  Batch   240  of    288.    Elapsed: 0:09:45.
  Batch   280  of    288.    Elapsed: 0:11:22.

  Average training loss: 0.18
  Training epcoh took: 0:11:42

Running Validation...
  Accuracy: 0.63
  Validation took: 0:00:30

======== Epoch 5 / 5 ========
Training...
  Batch    40  of    288.    Elapsed: 0:01:37.
  Batch    80  of    288.    Elapsed: 0:03:15.
  Batch   120  of    288.    Elapsed: 0:04:52.
  Batch   160  of    288.    Elapsed: 0:06:29.
  Batch   200  of    288.    Elapsed: 0:08:06.
  Batch   240  of    288.    Elapsed: 0:09:42.
  Batch   280  of    288.    Elapsed: 0:11:19.

  Average training loss: 0.08
  Training epcoh took: 0:11:38

Running Validation...
  Accuracy: 0.66
  Validation took: 0:00:29

Training complete!
```

Figure 7: Results of BERT for 5 Epochs

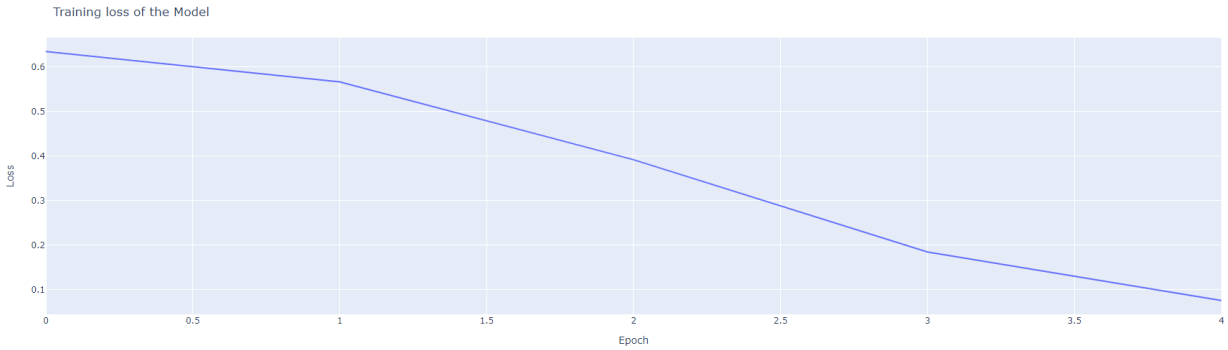

Figure 8: Training loss Vs Epochs plot

- Source Code is submitted as a separate IPYNB file.