FAKE NEWS DETECTION: THE STUDY OF TRADITIONAL AND MODERN APPROACH

A PREPRINT

Beenish Nazam
Student Number: 002327354
beenish.nazam@student.lut.fi

Sindhuja Chandrasekaran
Student Number: 002435644
sindhuja.chandrasekaran@student.lut.fi

Zarak Iftikhar
Student Number: 002294199
zarak.iftikhar@student.lut.fi

Himel
Student Number: 002448725
Aqemur.Himel@student.lut.fi

April 20, 2025

ABSTRACT

Misinformation through fake news is a growing concern in this era, making automated detection systems increasingly important. This project focuses on building an intelligent system to detect fake news using both traditional and modern machine learning approaches. We used a cleaned and sampled portion of the FakeNewsCorpus dataset to train a simple Naive Bayes model and an advanced Transformer-based model (DistilBERT). Both models were tested on a separate dataset, LIAR, to evaluate cross-domain performance. While the simple model served as a fast and effective baseline, the DistilBERT model achieved higher accuracy after fine-tuning. This report documents each stage of the process, including data cleaning, training, evaluation, and final observations.

Keywords FakeNewsCorpus dataset, LIAR dataset, Naive Bayes, DistilBERT

1 Data Processing

1.1 Data Retrieval and Sampling

The original FakeNewsCorpus dataset which downloaded from the link provided through instruction. It was very large in size with about about 27GB. Because it's too big to handle all at once. So we used about 10% of it. This smaller sample still gave us enough valuable information to study.

1.2 Tokenization

The text content was tokenized by separating it into lowercase words while removing punctuation. This step prepares the text for meaningful analysis by breaking it down into its smallest semantic units. We tokenized the text using nltk.RegexpTokenizer to remove punctuation and split the content into clean word tokens.

1.3 Removing Stopwords

Words like "the," "and," or "is" appear very often but don't add much meaning. Using NLTK's English stopword list, we filtered out common words that carry little semantic meaning .So, we removed them to focus on more useful words. We got the following results after removing the stopwords.

- Words before removing stopwords: 137,340
- Words after removing stopwords: 110,472

- Vocabulary got smaller by: about 19.6%

1.4 Reducing Word Variations (Stemming)

We used a method to reduce words to their root form. For example, "running," "ran," and "runs" all become "run." To unify word variations, we applied stemming using PorterStemmer.This helps group similar words.

- Words after stemming: 88,012
- Reduction after stemming: about 20.3%
- Total reduction from start to finish: about 35.9%

1.5 Key Observations

The final Sampled Dataset Size was 852909 rows which was 10% of the total datset.Before preprocessing Initial Vocabulary Size was 1116207 which was reduced to 969271 after preprocessing.

1.6 Method Justification

In Natural Language Processing (NLP), these steps stemming, stopword removal, and tokenization are critical for simplifying the text, concentrating on meaningful information, decreasing noise, and rising computational efficiency for subsequent modeling tasks.

2 Exploring the Data

2.1 How We Used the Data

The dataset was structured using a tabular format where each article is stored as a row, and columns include features such as content and type (label indicating reliability). This format is ideal for filtering, aggregating, and visualizing data properties efficiently.

2.2 Key observations

- **Before Cleaning**: most common frequent words were things like “the” and “and".
- **After cleaning**: we saw more useful frequent words like “trump", “clinton,” “email,” and “fbi". These are important because fake news often talks about politics.
- **Number of URLs**: The number of URLs present in the 10% of data was 181873.
- **Total no of dates and numbers**: The 10% data contains 43127 dates and 8334412 dates.

2.3 Train/Validation/Test Split

We took 80% of the data for training, 10% for the test, and 10% for validation.

2.4 Class Labels

We noticed that there were more articles labeled either fake or real (not balanced). This can be a problem when training a model and may need special handling.

2.5 Word Frequencies

We looked at the 100 most common words:

- Before cleaning: many unimportant function words.
- After cleaning: more meaningful content words showing patterns in the dataset.

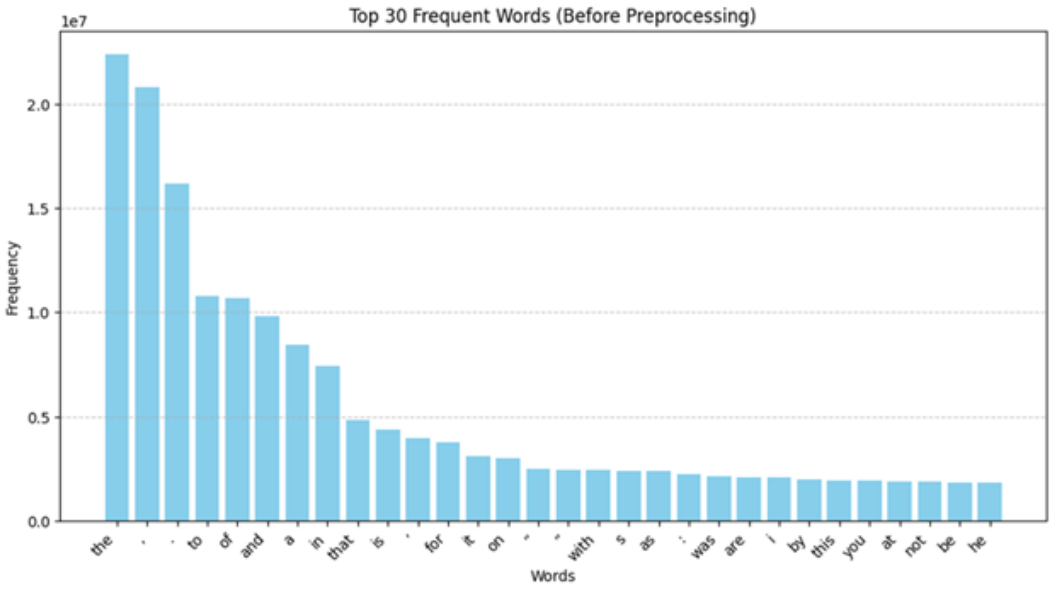

Figure 1: Top 30 Frequent Words before Preprocessing

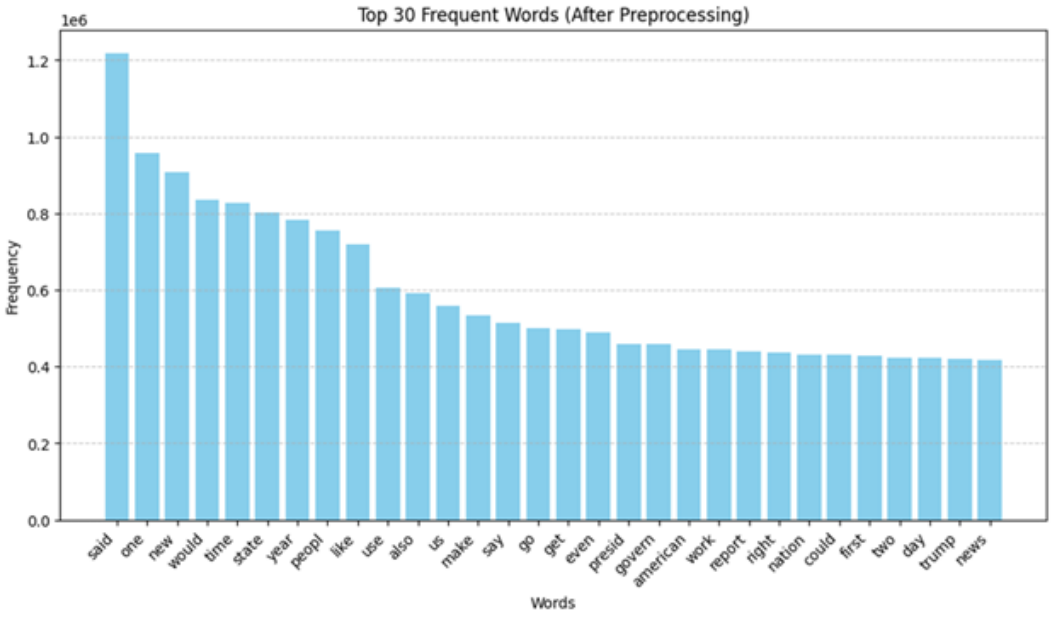

Figure 2: Top 30 Frequent Words after Preprocessing

3 Simple Model

3.1 Label Grouping

To simplify the classification task, we grouped all original labels into two main categories: 'fake' and 'reliable'. This makes it easier for the model to learn by creating a clear boundary between classes. Labels (e.g., "fake", "conspiracy", "political", "bias", "unreliable") were mapped to binary classes.

3.2 Baseline Results

Classification Report for Simple Baseline Model (TF-IDF + Naive Bayes)

	precision	recall	f1-score	support
fake	0.88	0.99	0.93	124302
reliable	0.96	0.55	0.70	38241
accuracy			0.89	162543
macro avg	0.92	0.77	0.82	162543
weighted avg	0.90	0.89	0.88	162543

Figure 3: Simple Model Results

We picked Multinomial Naive Bayes because it is simple, works well for text, and trains quickly. We used the Multinomial Naive Bayes (MNB) model with TF-IDF (Term Frequency-Inverse Document Frequency) vectorization features as our simple model. TF-IDF helps identify important words by reducing the weight of very common ones. It achieved 89% accuracy on the FakeNewsCorpus. Then saved the model to use for the liar dataset. This model was fast and easy to understand, making it a reasonable starting point due to its simplicity, speed, and interpretability.

3.3 Results After Metadata Inclusion

We added features like word count, character count, and average word length to see if metadata improved the model. Naive Bayes doesn't handle scaled numerical metadata well, making logistic regression a better fit for combining text and meta features. Hence, we used Logistic Regression to train a model with both text and metadata. The result was a high accuracy of 93%, but to keep the project focused, we chose to use only the main article content for all later tasks. After training, we tested the model and got the following results.

	precision	recall	f1-score	support
fake	0.94	0.97	0.95	62151
reliable	0.89	0.79	0.84	19120
accuracy			0.93	81271
macro avg	0.92	0.88	0.90	81271
weighted avg	0.93	0.93	0.93	81271

Figure 4: Simple Model with metadata

The model performed very well, with an overall accuracy of 93%. especially in distinguishing between fake and reliable articles. In practice, metadata varied widely and sometimes duplicated what text already says. However, We chose not to include them here, to keep our simple model focused on text. Hence for the rest of the project, we will focus only on using the main text (content - actual body of the news article), as required.

4 Advanced Model

To build a more powerful predictor, we implemented a Transformer-based model: DistilBERT, a lighter version of BERT from Hugging Face. It is pre-trained on large datasets and captures deep language patterns. Due to memory limits, we split the dataset into smaller parts for training. Based on the total number of rows in 10% of 27 gb and available ram, we calculated the recommended chunk and saved the advance model for the resulting for liar dataset.

```
Training Completed

 Advance Model Final Classification Report :
              precision    recall  f1-score   support

        fake       0.99      0.92      0.96      1481
   political       0.96      1.00      0.98      2999

    accuracy                           0.97      4480
   macro avg       0.98      0.96      0.97      4480
weighted avg       0.97      0.97      0.97      4480
```

Figure 5: Advanced Model Results

The model clearly outperformed the baseline, showing excellent generalization and better handling of both classes.

5 Evaluation

5.1 Comparison

The Naive Bayes model reached 89% accuracy, while DistilBERT performed better with 97% accuracy, precision, and recall.

Table 1: Test Set Performance(FakeNewsCorpus)

Model	Accuracy	Precision	Recall	F1
Naive Bayes (TF-IDF)	0.89	0.90	0.89	0.88
DistilBERT	0.97	0.97	0.97	0.97

5.2 Test on Liar Dataset

Both models were tested on the LIAR dataset to see how they performed on different data. After loading the liar data set, we explored the data then did label mapping. We loaded the trained models (simple and advanced trained models) without retraining to check how they generalize across domains. The text gives the main statement, content adds more details, type tells what kind of statement it is, and binary_label shows whether the statement is true or false for the computer to predict. Without these columns, the data would be unorganized and harder for the model to understand. The text gives the core information for prediction, making the task of detecting fake news much harder.

Cleaned and Binary Mapped LIAR Test Data:

	label	type	binary_label
0	true	reliable	1
1	false	fake	0
2	false	fake	0
3	half-true	fake	0
4	pants-fire	fake	0

Figure 6: Binary Mapping on LIAR Dataset.

Simple Model Evaluation on LIAR Dataset

	precision	recall	f1-score	support
fake	0.65	1.00	0.78	818
reliable	0.62	0.01	0.02	449
accuracy			0.65	1267
macro avg	0.64	0.50	0.40	1267
weighted avg	0.64	0.65	0.51	1267

Figure 7: Simple Model on LIAR Dataset.

Naive Bayes performed poorly on reliable articles. It over-predicted "fake" due to domain mismatch. This result shows overfitting to the FakeNewsCorpus domain. The model could not generalize to new language patterns.

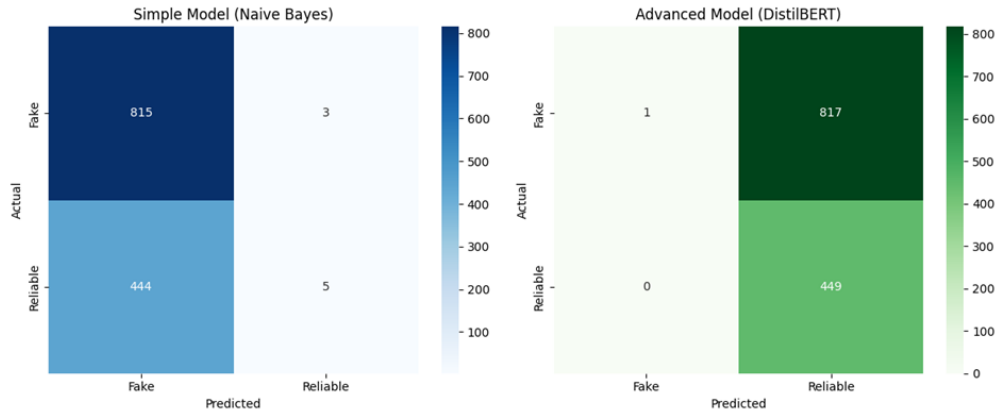

Figure 8: Simple Model vs Advanced Model.

Based on the confusion matrix, the simple model predicted most articles as fake, missing nearly all reliable ones (444 misclassified). The advanced model misclassified almost all fake news as reliable (817 errors), showing domain generalization issues.

To improve performance, we fine-tuned the model directly on the LIAR dataset using Hugging Face Trainer. Hence, the accuracy rose to 83%.

Advanced Model Evaluation on LIAR Dataset (DistilBERT)

	precision	recall	f1-score	support
fake	0.00	0.00	0.00	818
reliable	0.35	1.00	0.52	449
accuracy			0.35	1267
macro avg	0.18	0.50	0.26	1267
weighted avg	0.13	0.35	0.19	1267

Figure 9: Advanced Model on Liar Dataset

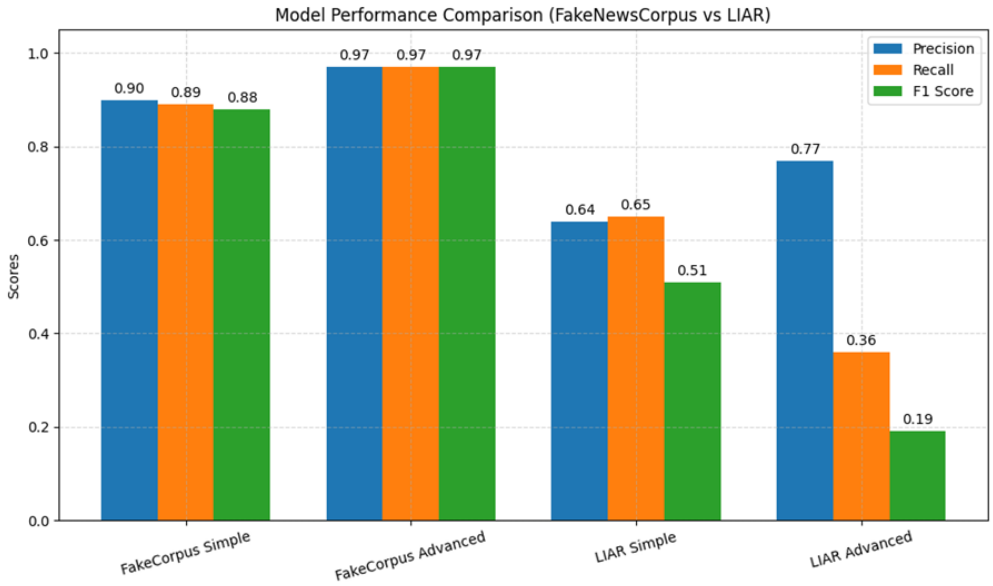

Figure 10: Performance Comparison

6 Conclusions

In fakenewscorpus dataset, Naive Bayes is a good baseline model with fast performance and 89% accuracy. DistilBERT performed significantly better in the original dataset with 97% accuracy. In liar dataset, The simple model had 65% accuracy and struggled with classifying reliable news. DistilBERT performed even worse initially, with 35% accuracy due to domain differences. However, both models had trouble when used on liar dataset, which shows the challenge of domain shift. Fine-tuning the transformer model helped fix this issue and increased the accuracy to 83%. What worked well was using transformer models and fine-tuning them to the dataset. Models trained on one dataset (like FakeNewsCorpus) did not work well when tested on a different dataset (like LIAR). This is called domain shift. Transformer models like DistilBERT can understand deeper language patterns compared to simple models. Fine-tuning the model on the new dataset i.e. on the liar dataset helped solve this issue and improved results. What didn't work was

grouping all types of fake news together and ignoring metadata. this hides differences between types of fake content, like: Satire (meant to be funny, not harmful), Clickbait (just for getting attention),Propaganda (intentionally misleading for political purposes).So, while this method helps training, it might reduce accuracy when detecting more complex forms of misinformation and Perform worse when faced with real-world data. So, while the binary approach helps training, it may make the model less flexible and accurate in more detailed or diverse situations. Future improvements could include detailed labeling, Instead of just "fake" or "real," use specific tags like satire, clickbait, etc. In short, Fake news detection can work very well on one dataset, but real-world success needs models that can handle new sources, writing styles, and platforms. To build a truly strong system, we should mix text, metadata, and social network signals (like who shared the article and when) to make detection much more reliable.

Group Member Contributions

All the group members equally contributed to every part of the project from scratch to an end.

The use of artificial intelligence tools in the present work

We hereby acknowledge that the use of ChatGPT to generate materials that were included within my final assessment in modified form. Some of the following were generated by AI initially :

- Error occurred while dividing the dataset, got to know on how to change from CPU and GPU and enabled GPU.
- To Understand Different Algorithms which acted as a base to select the models.
- We have Transformer and torch installed in the terminal of our VS Code, but it generated error, tried the ways like uninstalling , removing cache, reinstalling but still faced error hence used GPT which has given latest library name
- We have trained the models based on fakenewscorpus dataset and saved the model. Then we got the reference about the syntax form the GPT on how to reuse the model.
- For ordering and aligning the names in Latex documentation.