Fake News Detection NLP Approach

Popa Andrei-Ionut
andrei-ionut.popa@s.unibuc.ro

Semen Valentin-Ion
valentin-ion.semen@s.unibuc.ro

Stoica Elias-Valeriu
elias-valeriu.stoica@s.unibuc.ro

Toma Sabin-Sebastian
sabin-sebastian.toma@s.unibuc.ro

Abstract

This paper presents an in-depth exploration of different approaches to address fake news. Using natural language processing (NLP) techniques, we tested methods such as traditional machine learning, sentiment analysis, deep learning, and transformer-based models to enhance the detection and classification of fake news. Our research delves into the unique linguistic, structural, and contextual patterns that distinguish fake news from credible information.

1 Introduction

Fake news significantly undermines public trust and can sway public opinion, impacting elections, health decisions, and societal norms. The growing spread of misinformation has led to widespread consequences, such as:

- Misinformation influencing voter behavior (e.g., 2016 U.S. Elections).
- False claims during the COVID-19 pandemic (e.g., vaccine misinformation).

1.1 Methodologies Explored

To tackle this issue, we explored multiple methodologies, including traditional machine learning, sentiment analysis, deep learning models, and transformer-based approaches. These techniques aim to improve the accuracy of distinguishing between real and fake news by leveraging Natural Language Processing (NLP) techniques.

1.2 Goal of the Project

Our objective is to develop a robust system for detecting fake news by employing various NLP and Machine Learning techniques. Through this project, we aim to analyze the complexity of distinguishing fake news from real news, identifying their common features and characteristics that can indicate their type.

2 Related Work

The field of Natural Language processing has seen significant advancements in recent years, with a variety of approaches explored during the research. In this section, we review some of the contributions on this subject and its usage in the detection of fake news.

Devarajan et al. (2024) (Devarajan et al., 2024) proposed an innovative AI-assisted fake news detection framework using deep NLP techniques. Their model is structured into four layers: publisher, social media networking, edge, and cloud. The methodology includes four main steps: data acquisition, information retrieval, NLP-based feature extraction, and a deep learning classification model. The model integrates social and content-based features, such as publisher credibility, user activity, message context, and headlines, to classify news articles as real or fake.

Barrón-Cedeño et al. (2019) (Barrón-Cedeño et al., 2019) introduced Proppy, the first real-time propaganda detection system for online news. The system continuously monitors news sources, deduplicates, and clusters articles into events, and ranks them based on the likelihood of containing propagandistic content. It is trained using stylistic features derived from known propaganda sources and achieves state-of-the-art results on a standard dataset. Proppy aims to raise awareness about propaganda, thus mitigating its impact and contributing to the fight against disinformation.

By having these benchmarks, from our research, which offered us an idea of the diverse methodologies, we managed to set a stage for the development of different NLP models that are capable of detecting fake news.

3 Method

In the following sections we will describe the methods that were used including traditional machine learning, sentiment analysis, deep learning, and transformer-based models to enhance the detection and classification of fake news.

3.1 Dataset

The dataset used in this study is sourced from Kaggle, which provides a rich collection of news articles classified as either fake or real. This dataset serves as a crucial resource for understanding how misinformation and authentic information differ in terms of language, style, and sentiment.

Description: The original dataset contains 6,335 news articles, offering a balanced representation of fake and real news. The dataset includes the following features:

- **id**: A unique identifier for each news article.
- **title**: The headline of the article.
- **text**: The full body of the news article.
- **label**: A target variable that categorizes news as **FAKE (0)** or **REAL (1)**.

Issues: However, the raw dataset contains several challenges:

- **Missing Values:** Some rows lack data in critical fields, such as `text` or `title`.
- **Duplicate Entries:** Identical `title` or `text` entries across multiple rows, which can bias the results.

Statistics: After cleaning the dataset, we retained a total of 6,010 articles, with the following distribution:

- 3,025 fake news articles.
- 2,985 real news articles.

This balanced distribution makes the dataset suitable for both supervised learning and clustering experiments.

Relevance: Given its design, the dataset is expected to aid the model in identifying key linguistic and stylistic features that distinguish fake news from authentic news. These attributes are expected to be particularly relevant for training robust classification models.

3.2 Preprocessing

Preprocessing is a crucial step in preparing textual data for analysis, especially in Natural Language Processing (NLP) tasks. It ensures that the data is clean, consistent, and ready for input into machine learning models. Below, we outline the key preprocessing steps employed in this project:

All text is converted to lowercase to ensure uniformity, as most NLP models are case-insensitive. Punctuation marks are removed to focus on meaningful words and reduce noise in the data. Usernames (e.g., @username), links (e.g., https://...), and email addresses are removed, as they do not contribute to the understanding of the text's meaning. Text is split into individual words (tokens) using a tokenizer. This step breaks the text into manageable units for further processing. Common words like "the," "is," and "and" that carry little semantic meaning are removed to focus on the important content. Tokens that are not alphanumeric, such as special characters or numbers, are discarded to reduce noise. Each word is reduced to its base or dictionary form using a lemmatizer. For instance, words like "running" and "ran" are converted to their root form "run."

3.3 Traditional Machine Learning Approach

3.3.1 Techniques and Key Findings

The traditional machine learning approach focused on vectorizing text data using **Word2Vec**, which captured semantic relationships between words, providing dense and context-aware embeddings. This allowed for more accurate text representations, improving model performance.

The models trained included **Random Forest, Support Vector Machine (SVM), and Logistic Regression**. These classifiers were trained on Word2Vec vectors extracted from the dataset to predict the veracity of news articles.

Evaluation metrics included accuracy, precision, recall, F1-score, confusion matrix, ROC curve, and precision-recall curve to assess model performance. The classifiers achieved an **accuracy range of 0.90-0.92** across both validation and test sets.

3.3.2 Model Comparison: Random Forest vs SVM and Logistic Regression

Random Forest is an ensemble method that uses multiple decision trees for prediction. The model was configured with `200` estimators and a maximum depth of `20`.

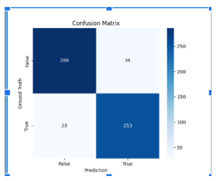
SVM and Logistic Regression models used a linear kernel (for SVM) and a linear regression approach for classification.

The models were trained using:

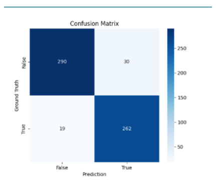
- **Random Forest:** `RandomForestClassifier(n_estimators=200, max_depth=20, random_state=42)`
- **SVM:** `SVC(kernel='linear', probability=True, random_state=42)`
- **Logistic Regression:** `LogisticRegression(max_iter=1000, random_state=42)`

3.3.3 Performance Evaluation

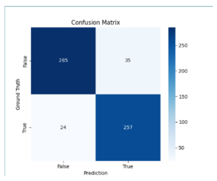
Confusion Matrix Analysis: The confusion matrices reveal that SVM had the best performance, achieving the lowest misclassification rate. Logistic Regression and Random Forest had similar results, with Logistic Regression slightly outperforming Random Forest in detecting positive cases.



(a) Random Forest



(b) SVM

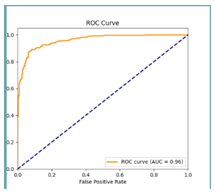


(c) Logistic Regression

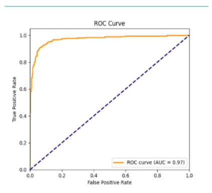
Figure 1: Confusion Matrices for Traditional ML Models

ROC Curve Analysis: All models exhibited strong discrimination capabilities, with **AUC scores of 0.96 for Random Forest and 0.97 for SVM and Logistic Regression**.

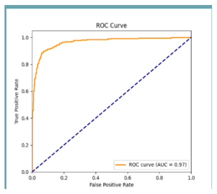
Precision-Recall Curve: All models maintained a good balance between precision and recall. SVM and Logistic Regression excelled in high precision for lower recall values, while Random Forest performed slightly better at higher recall levels.



(a) Random Forest

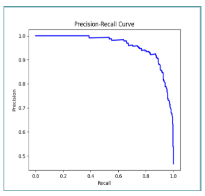


(b) SVM

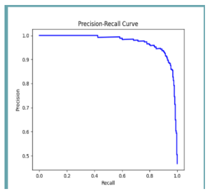


(c) Logistic Regression

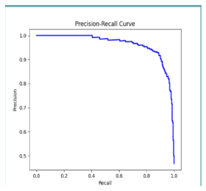
Figure 2: ROC Curves for Traditional ML Models



(a) Random Forest



(b) SVM



(c) Logistic Regression

Figure 3: Precision-Recall Curves for Traditional ML Models

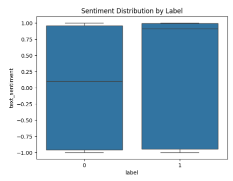
3.4 Sentiment Analysis Approach

- For the Sentiment Analysis method we tried predicting the genuineness of the news using a sentiment analysis approach.
- The goal was extracting sentiment features in order to be used for training a ML model on the selected dataset.
- We used the VADER(Valence Aware Dictionary and sEntiment Reasoner) Python library to obtain the sentiment score for the article titles and text content. The score ranges from -1.0 to 1.0, with -1.0 meaning negative sentiment, 0.0 neutral and 1.0 positive sentiment.
- We used the TextBlob library to obtain the

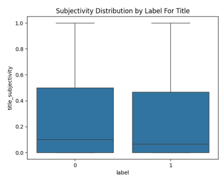
polarity and subjectivity score on the article titles and text content. The polarity and subjectivity score range from -1.0 to 1.0 similarly to the VADER score.

3.4.1 Sentiment Distribution By Label

The expected results for the sentiment scores would have been a more neutral score for the true label and a score closer to the extremes, either positive or negative for the false label, as fake news often display more extreme sentiments.
However, the scores seem to mostly overlap with the true news displaying a slightly more positive sentiment. The subjectivity distribution for the titles and text displayed a slightly more subjective sentiment for the fake news, while the polarity for both the text and titles followed the same trend as the sentiment score distribution.



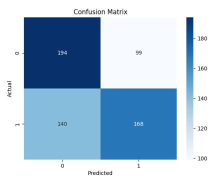
(a)



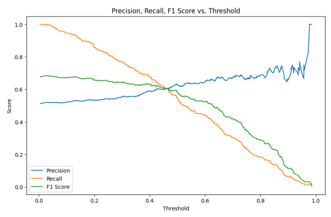
(b)

3.4.2 Training Model On Sentiment Features

An **XGBoost** model was then trained using the sentiment features and tested on about 10% of the dataset, with 10% of the dataset being used as validation. The model scored around 60% on the test dataset.



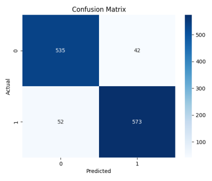
(c)



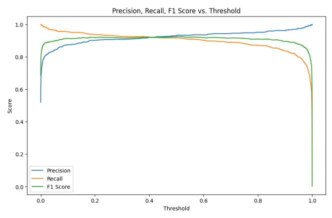
(d)

3.4.3 Sentiment Features + Embeddings

For comparison, the **XGBoost** model was then trained on the embeddings obtained using **Word2Vec** along with the sentiment features. The model scored approximately 92% accuracy on the test dataset.



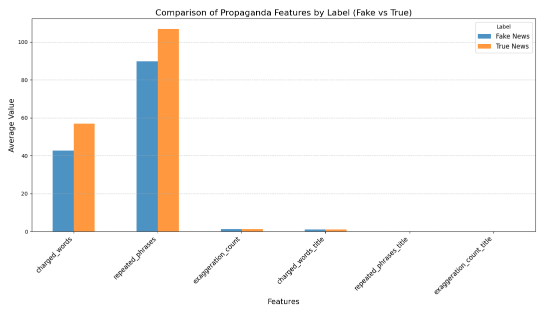
(e)



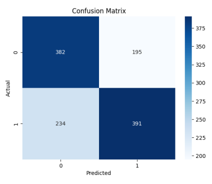
(f)

3.4.4 Improving Sentiment Only Approach - Adding Propaganda

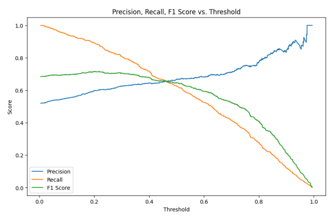
In order to improve the performance of the sentiment only approach, we tried adding propaganda related feature into the training process. The features that were extracted were the amount of charged words (words that are in the opinion lexicon positive and negative nltk corpus), repeated phrases and exaggerations (words like always, never etc). The expected outcome for this analysis was that the fake news would have higher propaganda features, however in actuality, the reverse happened, although slightly. However, these added features helped us achieve the 64% precision in the test data set.



(g)



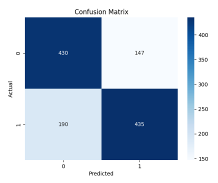
(h)



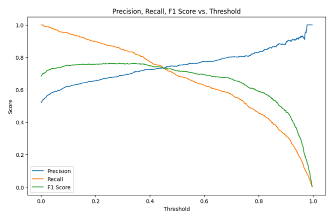
(i)

3.4.5 Improving Sentiment Only Approach - Additional Sentiment Features

The next method that we attempted in order to improve the performance of the model trained only on the sentiment features was adding additional features. Some of the features that were added include calculating sentiment statistics at sentence(mean, variance, minimum, maximum), capturing the first and last sentiment of the texts/titles, calculating the negative/positive/neutral ratios at word level for the titles/texts, computing the number of sentiment shifts, capturing the sentiment by part-of-speech (adverb, adjective, noun, verb), calculating the sentiment by named entity, the number of extreme sentiment words (greater than 0.7 score), calculating the sentiment trajectory by obtaining the proportion of which the sentiment increases after each sentence, repetition ratio, punctuation density, indexes for reading ease, exclamation and question counts, etc. In addition the model was changed to a **RandomForestClassifier**. With these additions the model was capable of obtaining 72% on the test dataset.



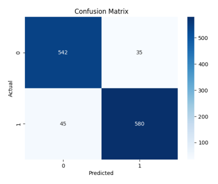
(j)



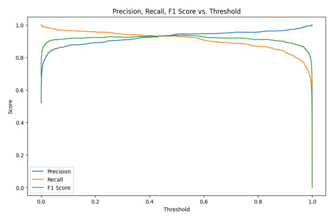
(k)

3.4.6 Improving Sentiment Only Approach - Testing With Embeddings

Adding back the embedding features, we obtained about 93% accuracy on the test dataset.



(l)



(m)

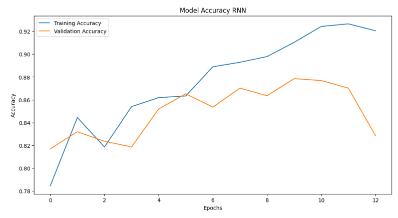
While useful, the sentiment features cannot predict with a very high accuracy on their own, given the overlap between the true and fake labels sentiment distribution.However, an accuracy of almost 73% was obtained by only using various sentiment features and it could be improve by using a larger dataset or by calculating more features in order to be used for training.

3.5 Deep Learning

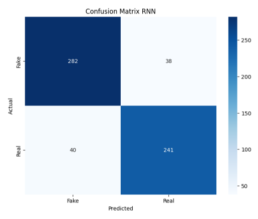
3.5.1 Model Architecture and Performance Metrics - RNN

The architecture utilized Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) cells to capture temporal dependencies in text data with ReLU and sigmoid activations to learn features. The model leveraged pre-trained Word2Vec embeddings to enhance text representation, followed by Dense Layers for classification. Performance metrics included accuracy, precision, recall, and F1-score, with results showing improvements over traditional approaches. Challenges included overfitting due to limited training data, requiring techniques like dropout and regularization to improve generalization. The training process utilized advanced optimization techniques, such as the AdamW optimizer with weight decay and learning rate, alongside binary cross-entropy loss function, early stopping, and model checkpointing.

Model Accuracy: ~ 0.8752



(n)

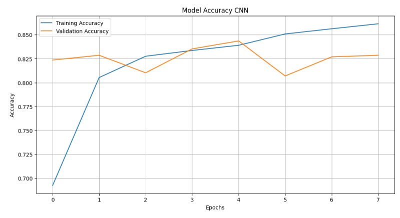


(o)

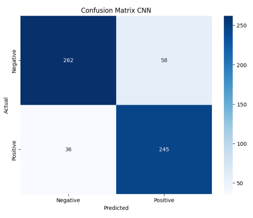
3.5.2 Model Architecture and Performance Metrics - CNN

The architecture utilized Convolutional Neural Networks (CNNs) with layers such as convolution, max pooling, dropout, and fully connected layers, using ReLU and sigmoid activations to learn features and prevent overfitting. The model leveraged pre-trained Word2Vec embeddings to enhance text representation. Challenges included handling class imbalance and optimizing hyperparameters through validation and performance monitoring. Evaluation metrics included accuracy, precision, recall, and F1-score, supported by confusion matrices and ROC curve analysis.

Model Accuracy: ~ 0.8469



(p)

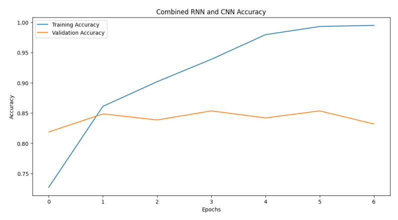


(q)

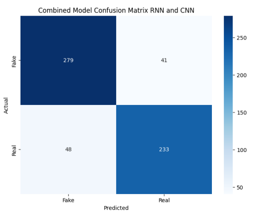
3.5.3 Model Architecture and Performance Metrics - RNN & CNN Combined

This architecture combined RNN and CNN branches using pre-trained Word2Vec embeddings, where the LSTM-based RNN captured sequential features, and the CNN captured spatial features from text. The outputs of the RNN and CNN branches were concatenated, followed by a fully connected layer and dropout to process the combined features and prevent overfitting. After training, the model performance was evaluated using accuracy, precision, recall, and F1-score, providing insights into the ability to distinguish between fake and real text.

Model Accuracy: ~ 0.8402



(r)



(s)

3.5.4 Key Observations from Accuracy Analysis

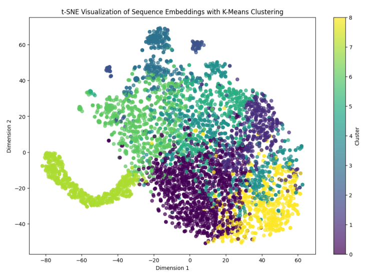
- **RNN:** Effective at learning from the training set but prone to overfitting after extended epochs.
- **CNN:** Demonstrated better generalization and validation performance, though training accuracy was slightly lower than RNN.
- **Combined Model:** While excelling at training data, overfitting limited its effectiveness for validation data.

3.5.5 Confusion Matrix Analysis

- **RNN:** Performs better in identifying negative instances (lower false positives) but misses slightly more positive instances compared to CNN.
- **CNN:** Slightly more false positives, indicating a tendency to over-predict the positive class.
- **Combined Model:** Balanced predictions but slightly worse in detecting positive instances compared to individual models.

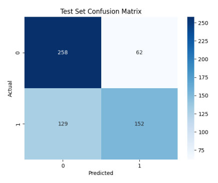
3.5.6 Unsupervised Learning - RNN

The t-SNE visualization reveals distinct clusters of data points, indicating that the sequence embeddings effectively group similar instances together. However, some overlap between clusters suggests that the embeddings or the clustering algorithm (e.g., K-Means) could benefit from further refinement to improve separation and reduce ambiguity in the group boundaries.



(t)

The confusion matrix demonstrates moderate performance, with a strong ability to correctly predict negative cases (high true negatives: 258) but struggles with misclassifying positive cases (relatively high false negatives: 129). This indicates the model might be biased toward the negative class, potentially requiring adjustments like better class balancing, feature engineering, or hyperparameter tuning.



(u)

Model Accuracy: ~ 0.6279

3.6 Transformer Models Approach

3.6.1 Introduction to Transformers

Transformers are powerful NLP models that can understand the meaning and context of words better than previous models. They work well with large amounts of text data and have been widely used for tasks like translation, text generation, and fake news detection. In this project, we use pre-trained transformer models that are fine-tuned specifically for identifying fake news.

3.6.2 Utilization of Pre-trained Transformers

Instead of training models from scratch, we use pre-trained transformer models that have already learned language patterns from large datasets. These models are then fine-tuned on fake news datasets, helping them recognize misleading information more effectively. This approach saves time and improves accuracy.

3.6.3 Overview of Selected Transformer Models

We experiment with four transformer models, each having unique features:

- BERT (Bidirectional Encoder Representations from Transformers): A model trained on massive text datasets that understands context by analyzing words in both directions.
- DistilBERT: A smaller and faster version of BERT that keeps most of its accuracy while improving efficiency.
- XLNet: A model that improves on BERT by capturing better dependencies between words.
- RoBERTa (A Robustly Optimized BERT Pretraining Approach): A more advanced version of BERT trained with more data and improved training techniques for better performance.

3.6.4 Objective

We utilized the pre-trained transformer model to classify news articles as **Fake** or **Real**. After training, we evaluated the model's performance using various metrics, including **accuracy, precision, recall, F1-score**, and a **confusion matrix** to analyze misclassifications.

3.6.5 Data Preprocessing

Before training, we processed the data with the following steps:

- **Merging Columns:** We combined the *title* and *text* columns into a single input field called *content* to provide the model with more context.
- **Removing Duplicates and Empty Entries:** This ensures redundant or incomplete data does not affect performance.
- **Label Encoding:** We assigned numerical labels to the target classes:
  - **Fake** $\rightarrow 0$
  - **Real** $\rightarrow 1$

3.6.6 Tokenization

To convert text into structured input, we used **BertTokenizer**. To ensure a uniform input format, we applied:

- **Truncation:** Cutting off text that exceeds the model's maximum length.
- **Padding:** Adding extra tokens when the text is shorter than required.

The **maximum token length** was set to **512 tokens**, as supported by transformer models.

3.6.7 Model and Training

Each transformer model was fine-tuned and optimized using:

- **AdamW Optimizer** – A weight decay variant of Adam that improves stability.
- **Learning Rate Scheduling** – Adjusts learning rate dynamically for better training.
- **Loss Function: CrossEntropyLoss** – Measures prediction accuracy.
- **Training Duration:** We trained the model for **3 epochs**, tracking validation accuracy and training loss at each step.

3.6.8 Evaluation

To analyze performance, we used:

- **Softmax Probabilities** – To interpret prediction confidence.
- **Matplotlib & NumPy** – To visualize accuracy, loss curves, and confusion matrices.

These insights helped us understand where the model struggled and how well it distinguished between real and fake news.

3.6.9 Accuracy

All models performed exceptionally well in terms of accuracy, with **RoBERTa** achieving the highest accuracy of **99.5%**, closely followed by **BERT (98.0%)**, **XLNet (98.8%)**, and **DistilBERT (97.67%)**. The high accuracy across models suggests that transformer-based architectures are highly effective for fake news classification.

3.6.10 Precision & Recall

Precision scores show a similar trend to accuracy, with **RoBERTa** again leading at **99.5%**, indicating its strong ability to minimize false positives. **DistilBERT**, while slightly lower, demonstrated that its lightweight architecture still maintains reliable performance.

Recall scores highlight how well the models identify actual positive cases. **RoBERTa** achieved the highest recall at **99.5%**, while **XLNet** had a lower recall (**93.0%**), suggesting that it may miss some true positive cases. This difference highlights the trade-off between precision and recall across models.

3.6.11 Efficiency

DistilBERT is the most efficient model, reflecting its lightweight architecture and significantly shorter training time of just **7 minutes**. In contrast, **RoBERTa**, despite being the most accurate, had the lowest efficiency due to its long training time of **31.5 minutes**. This trade-off is important when considering real-world applications that require both accuracy and efficiency.

3.6.12 Performance and Efficiency Visualization

To better understand the trade-offs between accuracy, precision, recall, F1-score, and efficiency, we present a comparative graph below. This visualization helps to illustrate each model's strengths and weaknesses.

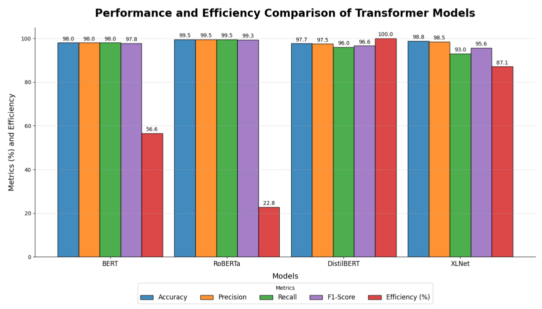
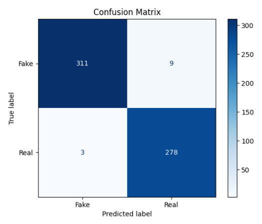


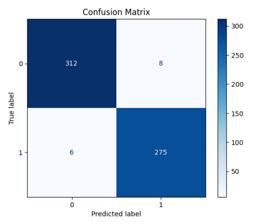
Figure 4: Performance and Efficiency Comparison of Transformer Models

3.6.13 Comparison of Confusion Matrices

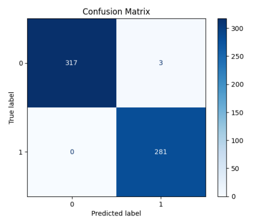
To evaluate the performance of each transformer model, we present their confusion matrices below. These matrices show the number of correctly and incorrectly classified fake and real news articles.



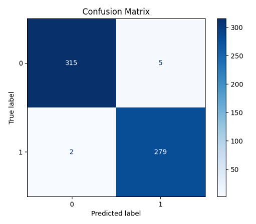
(a) BERT



(b) DistilBERT



(c) RoBERTa



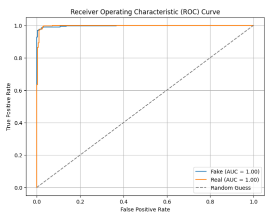
(d) XLNet

Figure 5: Comparison of Confusion Matrices for Different Transformer Models

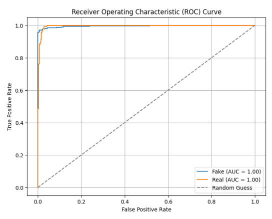
3.6.14 Comparison of ROC Curve Graphs

The Receiver Operating Characteristic (ROC) curves illustrate the classification performance of each transformer model. The Area Under the Curve (AUC) values for all models are **1.00**, indicating perfect performance in distinguishing between fake and real news during testing.

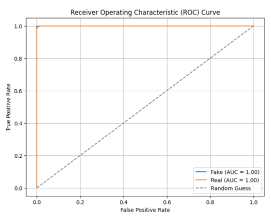
These results suggest that all models are highly effective in their classification task, achieving optimal trade-offs between the **true positive rate** and **false positive rate**. The ROC curves confirm that our models can successfully differentiate between real and fake news with near-perfect accuracy.



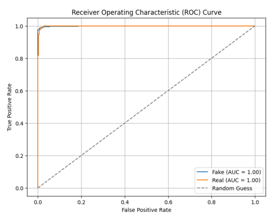
(a) BERT



(b) DistilBERT



(c) RoBERTa



(d) XLNet

Figure 6: Comparison of ROC Curves for Different Transformer Models

4 Future Work

- **Eliminating Language Barriers:** Current models are primarily trained on English datasets. Expanding the dataset to include multilingual news articles can significantly improve performance across different languages.
- **Browser Extension for Real-Time News Verification:** Develop a user-friendly browser extension that utilizes trained models to classify news articles as fake or true. Users could input a news headline or article directly, and the extension would provide an instant credibility score along with an explanation for its classification. This tool could serve as an accessible and practical way to combat misinformation at the user level.

5 Conclusion

The project showcased the effectiveness of diverse NLP techniques in addressing the challenge of fake news detection. Traditional machine learning, sentiment analysis, deep learning, and transformer models each brought unique strengths, enabling a comprehensive approach to classification. While useful, the sentiment features cannot predict with a very high accuracy on their own, given the overlap between the true and fake labels sentiment distribution.However, an accuracy of almost 73% was obtained by only using various sentiment features and it could be improve by using a larger dataset or by calculating more features in order to be used for training. Future work can explore unsupervised methods, enhancing detection capabilities with minimal labeled data, and adapting models for multilingual datasets to broaden applicability across different languages and cultures.

References

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: A system to unmask propaganda in online news.

Ganesh Gopal Devarajan, Senthil Murugan Nagarajan, Sardar Irfanullah Amanullah, S. A. Sahaaya Arul Mary, and Ali Kashif Bashir. 2024. Ai-assisted deep nlp-based approach for prediction of fake news from social media users. *IEEE Transactions on Computational Social Systems*, 11(4):4975–4985.