Natural Language Processing Techniques for Distinguishing Between Fake and Real News: A Comparative Study of Machine Learning Models

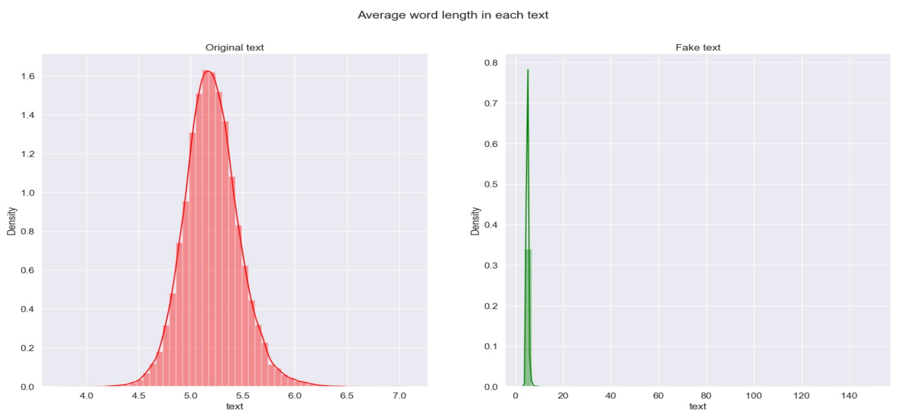
Group Members:

- Shivani Pore (G01333617)
- Shubham Milind Kasar (G01384017)
- Aniket Anil Raut (G01387118)

Dataset:

The dataset used in this study was sourced from Kaggle and consisted of a combination of both fake and real news. The news articles were extracted from Reuters and contained various columns, including title, text, subject, and date. The text column in the dataset contained different news articles.

Are there any patterns or trends in the use of language or writing style that distinguish "fake" news articles from "real" ones?



From the above figure, we successfully identified the pattern relating to the length of the words in fake news articles and real news articles. The density of words in real news articles is higher and more spread out compared to the fake ones.

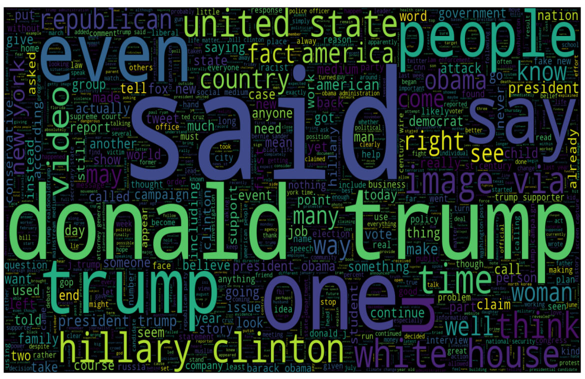
Data Preprocessing:

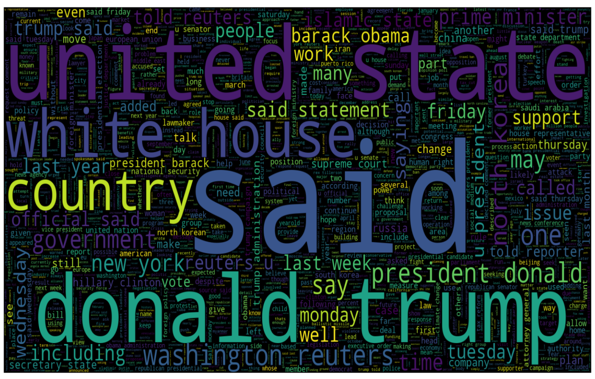
For data preprocessing and cleaning, several natural language processing (NLP) techniques were utilized, including stopword removal, stemming, lemmatization, and Part-of-Speech (PoS) tagging. These techniques were applied to the dataset to extract important words from the sentences, which were then used to analyze the notable differences between real and fake news. Tokenization was also performed to further enhance the analysis of the data.

Method:

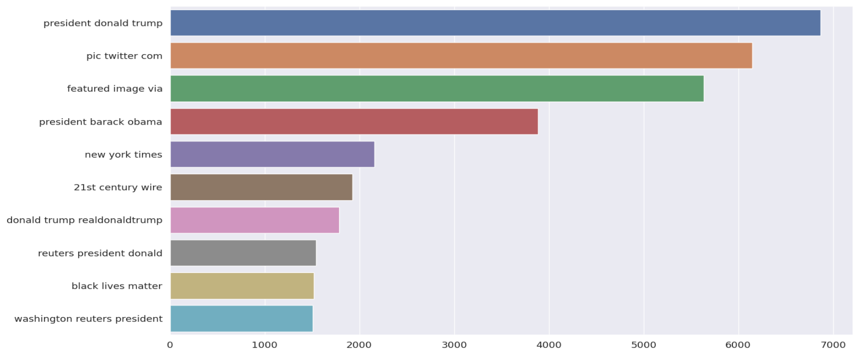
Are there certain keywords or topics that are more common in "fake" news articles?

We defined a function get_corpus() where we got different corpus and using the counter we got the various repeating corpus, to understand the keywords that were used in fake and real news articles and how many times they were repeated. From the below plot, we can see the most repeating keywords.





We created a bag of words and plotted graphs for various n-grams, to understand the semantics between the word and the word above and below it. From the below graph we can see that the trigram “president donald trump” has been used in more than 6500 news.
To get the important features we mainly used 3 techniques that are the bag of words (Count Vector) and frequency weights with various n-grams (TF-IDF) and word embeddings. For performance evaluation of different models we used confusion matrix, precision recall and f1 score.



Bag of words Technique:
For the Bag of Words approach, the dataset was tokenized and a count vector of the frequency of different words was created. The count vectorizer was used to extract features, with a maximum of 14 features.

Frequency weights with various n-grams:
To experiment with another technique, frequency weights with various n-grams were used for feature selection. Features were extracted using the TF-IDF vectorizer with an n-gram range from 1-3.

Word Embedding:
For the Word Embedding technique, pre-trained word embeddings were used to represent words in a high-dimensional vector space that captures semantic relationships between words. The embeddings were used to initialize the embedding layer of a neural network for natural language processing tasks. The pre-trained embeddings were obtained from a corpus of Twitter data using the GloVe algorithm. Before applying pre-trained word embedding, Tokenizer() was used to limit the vocabulary to 10,000 words and pad them to 300 to avoid overfitting.

Classifiers:

- MultinomialNB:
  A MultinomialNB classifier from sklearn.naive_bayes library was used for both the Bag of Words and TF-IDF techniques, and the features extracted from the count vector and TF-IDF vector were passed to it.

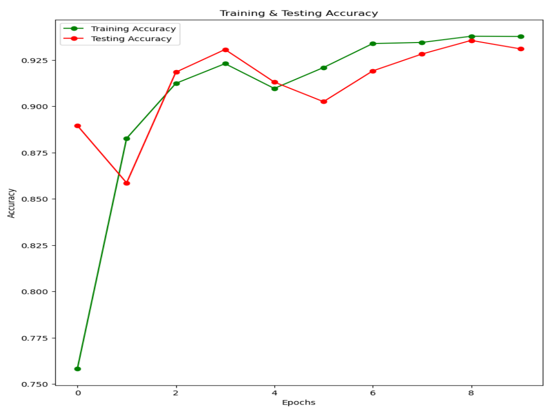
- LSTM:
  For the LSTM model, a neural network with two layers, a dropout of 0.7, and activation functions such as ‘sigmoid’ and ‘relu’ were used. The model was trained for 10 epochs on a batch size of 128 and a learning rate of 10e-3.

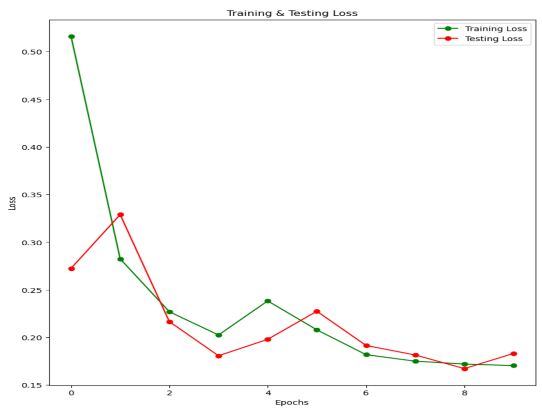
Following is the Analysis of classifiers:
The following table presents the evaluation metrics (precision, recall, F1 score, and accuracy) for three different models (Naive Bayes with bag of words, Naive Bayes with TF-IDF, and LSTM with word embeddings) on a binary classification task of distinguishing true news from fake news. The table also includes the performance of each model on the two classes (true and fake).

	Precision		**Recall**		**F1 Score**		**Accuracy**
	True	**Fake**	**True**	**Fake**	**True**	**Fake**	
NB (Bag of words)	0.77	0.82	0.82	0.77	0.79	0.80	0.80
NB(TF-IDF)	0.96	0.95	0.95	0.96	0.96	0.96	0.95
LSTM (word embedding)	0.97	0.90	0.88	0.98	0.92	0.94	0.93

Both NB (Bag of words) and NB (TF-IDF) models have similar performance across all metrics, with precision, recall, F1 score, and accuracy all around 0.8-0.95. The NB (TF-IDF) model performs slightly better than the NB (Bag of words) model in terms of precision, recall, and F1 score.
The LSTM (word embedding) model outperforms both NB models in terms of precision and accuracy, with precision of 0.97 and accuracy of 0.98. However, it has a lower recall of 0.90, which means that it may miss some True cases. As a result, its F1 score is lower than the NB (TF-IDF) model's F1 score.

Graphs:

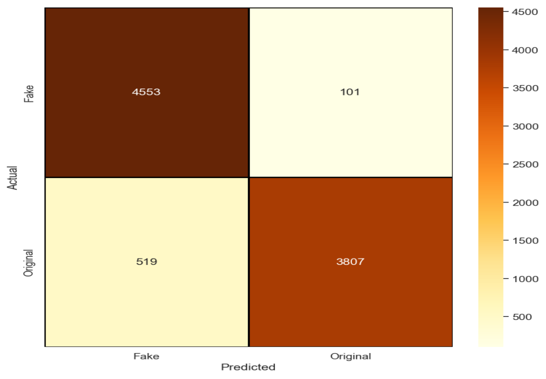


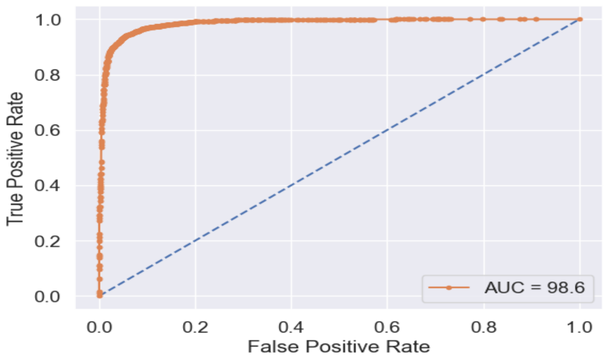


<u>Training Accuracy and Testing Accuracy:</u> The graph for accuracy shows that the model's training accuracy consistently increases over the 10 epochs, and the testing accuracy also increases but seems to plateau around the 5th epoch. The final testing accuracy is around 93%. This suggests that the model is performing well on the training data and is generalizing well to the testing data.

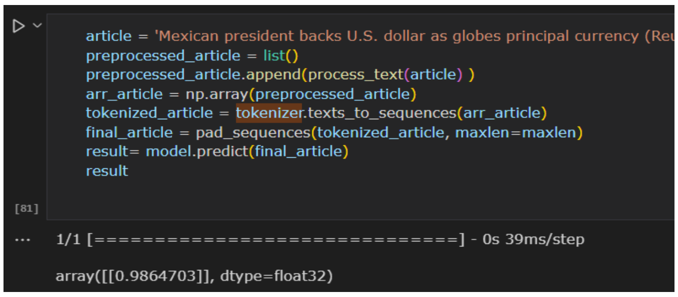
<u>Training Loss and Testing Loss:</u> The graph for loss shows that the model's training loss decreases rapidly in the first few epochs and then plateaus around the 6th epoch. The testing loss follows a similar pattern but does not plateau as much. The final testing loss is around 0.23. This suggests that the model is learning from the training data and is not overfitting too much to the testing data.
Overall, the graphs suggest that the LSTM model is performing well and is not overfitting to the training data.

From the confusion matrix, we can see that the model is performing well on the majority of the samples, as the true positives and true negatives are much higher than the false positives and false negatives.The AUC-ROC score for the model is 98.6%, which indicates that the model has a high degree of accuracy in distinguishing between the positive and negative classes. The ROC curve shows the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis for different thresholds of the model's predicted probabilities. The curve is close to the top-left corner of the plot, indicating that the model has a high TPR and a low FPR, which is a good performance. Overall, the high AUC-ROC score and the shape of the ROC curve suggest that the model is performing well on the classification task and has a high degree of accuracy in distinguishing between the positive and negative classes.





Testing:
Can the model accurately classify news articles from different sources, and are there any sources that are particularly difficult to classify?
We randomly selected an article from Reuters and fed it to our best model, which is the LSTM. This model has an accuracy of 93%, and it correctly predicted the result. During the testing phase, we fed the model with articles from various sources, and it encountered difficulty in accurately classifying them due to the varying writing styles. As a result, the model was predicting some genuine articles as fake. To enhance the model's efficacy, we suggest that future improvements could include training the model on a more diverse range of genuine and fake articles with different writing styles. This training approach would likely enhance the model's ability to accurately classify articles from various sources.



```
article = 'Mexican president backs U.S. dollar as globes principal currency (Reu
preprocessed_article = list()
preprocessed_article.append(process_text(article) )
arr_article = np.array(preprocessed_article)
tokenized_article = tokenizer.texts_to_sequences(arr_article)
final_article = pad_sequences(tokenized_article, maxlen=maxlen)
result= model.predict(final_article)
result
```

```
[81]
1/1 [==============================] - 0s 39ms/step

array([[0.9864703]], dtype=float32)
```

Conclusion:
In conclusion, this project aimed to distinguish between true and fake news using various natural language processing techniques and machine learning models. The dataset used consisted of news articles from Reuters, which were preprocessed using NLP techniques such as stopword removal, stemming, lemmatization, and PoS tagging. The bag of words, TF-IDF, and word embedding techniques were used to extract features, and three classifiers were trained: Naive Bayes with bag of words, Naive Bayes with TF-IDF, and LSTM with word embeddings. The evaluation of these classifiers showed that the LSTM model with word embeddings outperformed the Naive Bayes models on most metrics, indicating that deep learning techniques can be highly effective for natural language processing tasks. The Naive Bayes model with TF-IDF performed significantly better than the Naive Bayes model with bag of words on all metrics, indicating that the TF-IDF technique is more effective at feature selection than the bag of words technique.

Video Link :
https://gmuedu-my.sharepoint.com/:v:/g/personal/skasar_gmu_edu/EdH3H9DNgUlOnYuNOcrFF9gBFRUZ1Tc939vGIpFFbvYSQw?e=QrhGPG

References :

1. https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a
2. https://www.nature.com/articles/s41598-021-01460-7
3. https://edumunozsala.github.io/BlogEms/jupyter/nlp/classification/embeddings/python/2020/08/15/Intro_NLP_WordEmbeddings_Classification.html
4. https://www.analyticsvidhya.com/blog/2021/06/natural-language-processing-sentiment-analysis-using-lstm/