Fake and Real news classification

Group Final Term Project Report

Meghana Gantla

Kohisha Aruganti

DATS 6312 - NLP for Data science

Instructor - Amir Jafari

Table of Contents

Introduction

Fake news has become a major problem in the digital era, especially with the increasing use of social media platforms. The spread of fake news can cause significant harm to individuals, organizations, and even nations. Therefore, we decided to develop an accurate and reliable fake news classification model that can distinguish between real and fake news. The scope of our project is to employ both classical and non-classical machine learning models to accurately classify the between True and Fake.

Dataset Description

We will use the "Fake and real news dataset" available on Kaggle. This dataset contains 44898 news articles, of which 21417 are labeled as real news and 23481 as fake news. Each article has a 'title', 'text', 'subject', and 'date'. We combined true.csv and fake.csv into a single csv file and added a column with 0 and 1 values to be our target column.

The articles in True.csv are collected from Reuters.com and the articles in fake.csv are collected from a few unreliable sources and also wikipedia. We also decided to combine the ’title’ and ‘text’ columns in the dataset to create the final text column used for all of the analysis. We also have a column ‘subject’ showing the category of news this article belongs to. We performed EDA to better understand the distribution of each column.
https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

Data Preprocessing

We performed several preprocessing steps on the data as follows,

- We removed all the URLs
- Applied lower casing
- Removed Contractions, and punctuations
- Removed stop words and
- Applied Lemmatization.

We have a separate function for each of these steps in our code and they deliver results as expected. Once the data goes through all of these functions it is normalized to the expected extent. This has helped reduce the number of unique tokens in the dataset and therefore improved computation load.

Exploratory Data Analysis

We decided to perform Exploratory data analysis on the data to better understand the distribution of the data and also understand the relationships between some of the columns. To begin with, we checked for the counts of fake and real news in the dataset to make sure there is no need for upsampling or downsampling. Here are the results for that and as we can see there was no need for any upsampling/downsampling.

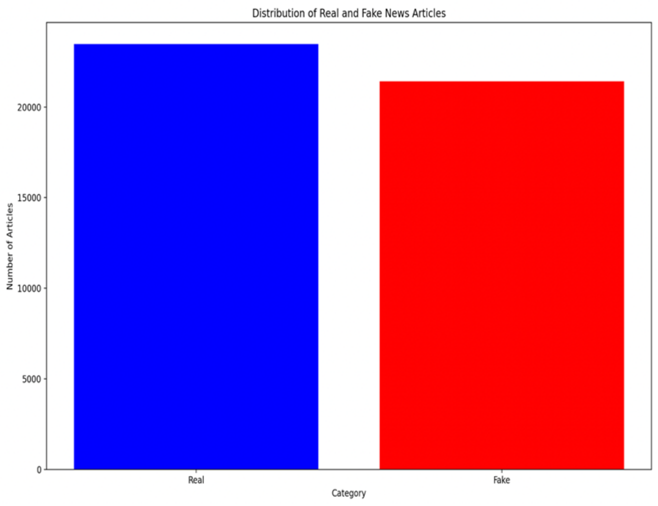


Figure 1: Real vs Fake news counts

With the help of our next visualization, we can better understand the distribution of the length of the articles in the dataset. We can see that the highest frequency is around 30000 in this plot which means that is the most common size of article in this dataset.

Further, we wanted to check how the subject of the articles are tagged for both fake and real articles. Here we found something really interesting. The real articles only had two of the given list of subjects as their values and the rest are all present only for fake articles. If we were to include this column in our analysis, the results will not be accurate since subjects become the easy and the best way to classify the articles. In the results we can see that only ‘worldnews’ and ‘politicsnews’ subjects in the data are real articles and the rest all belong to fake articles. We dropped this column here and did not include it in the analysis. We also dropped the ‘Date’ column because we felt it is irrelevant how old the article is for it to be real or fake.

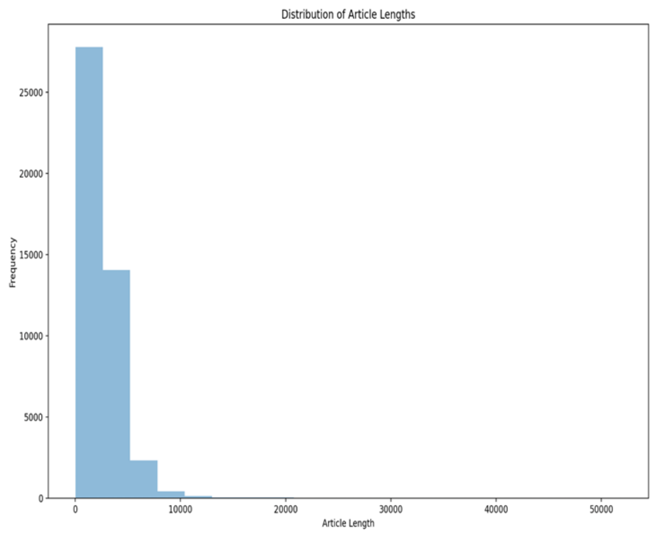


Figure 2: Distribution of Article Length

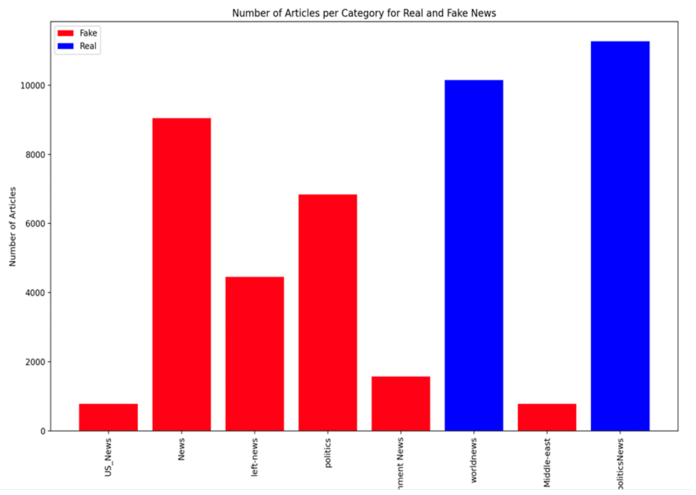


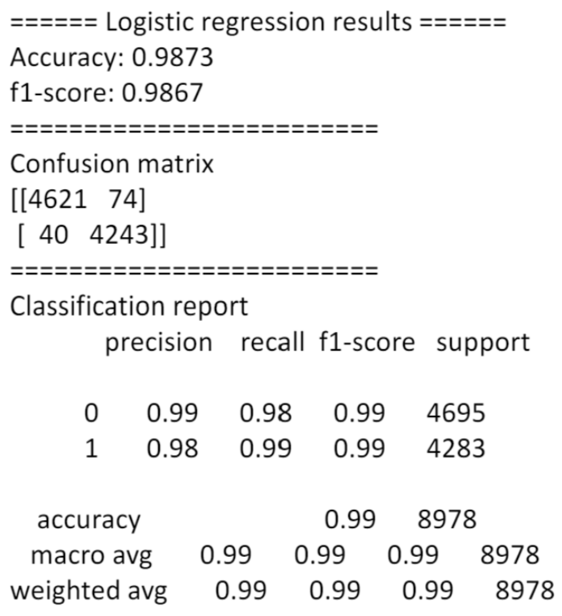
Figure 3: Number of articles by subject

Modeling

For modeling, we decided to employ both classical and non-classical machine learning models. In classical models, I have worked on Linear Regression, and Naive Bayes classifier. In the non-classical models, I have worked on the DistilBERT transformer model.

Logistic Regression

I performed logistic regression on the preprocessed data and it has yielded me with really good results as below. The accuracy score is fairly high and the confusion matrix shows how the values are predicted correctly and how many of them were predicted wrong.



```
====== Logistic regression results ======
Accuracy: 0.9873
f1-score: 0.9867
========================
Confusion matrix
[[4621   74]
 [  40  4243]]
========================
Classification report
             precision    recall  f1-score   support

           0      0.99      0.98      0.99      4695
           1      0.98      0.99      0.99      4283

    accuracy                          0.99      8978
   macro avg      0.99      0.99      0.99      8978
weighted avg      0.99      0.99      0.99      8978
```

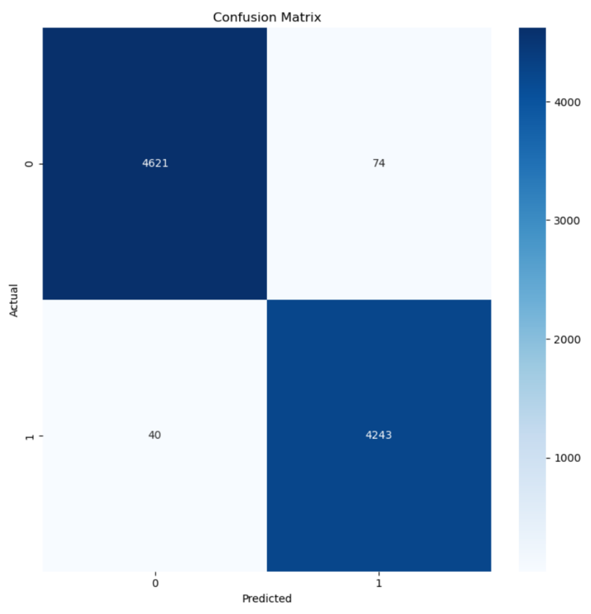
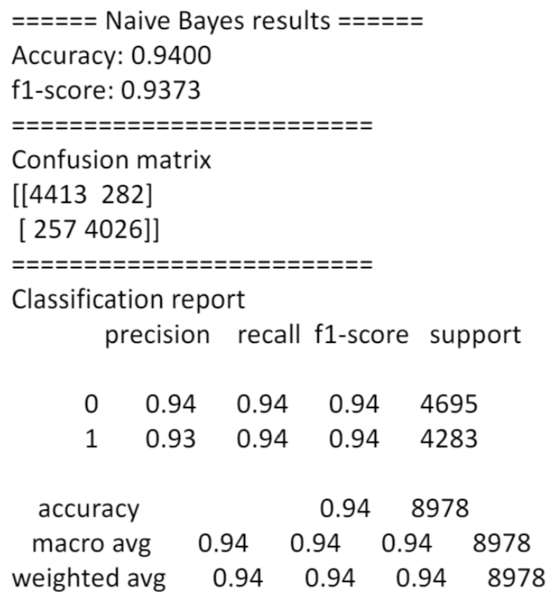


Figure 4 & 5: Logistic regression results

Naive Bayes Classification

After Logistic regression, I wanted to try out another classical machine learning model and ended up trying Naive bayes. However, it did not yield better results than logistic regression. Here we can see the results for the model and also the confusion matrix which is slightly worse than the previous model.



```
====== Naive Bayes results ======
Accuracy: 0.9400
f1-score: 0.9373
=========================
Confusion matrix
[[4413  282]
 [ 257 4026]]
=========================
Classification report
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      4695
           1       0.93      0.94      0.94      4283

    accuracy                           0.94      8978
   macro avg       0.94      0.94      0.94      8978
weighted avg       0.94      0.94      0.94      8978
```

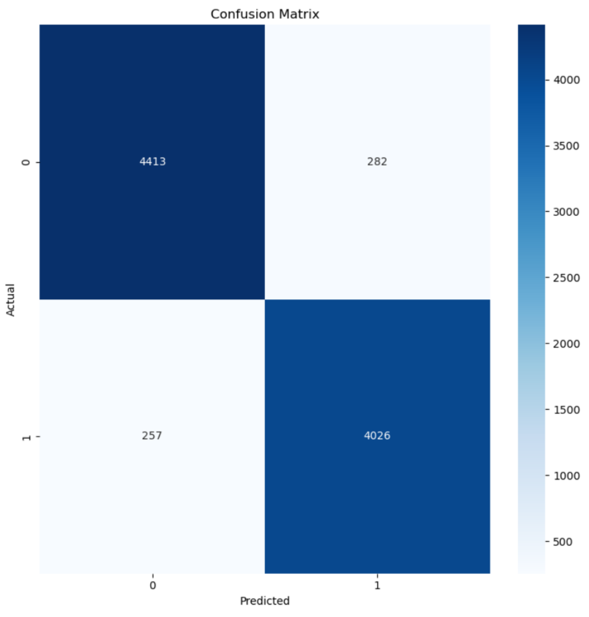


Figure 6 & 7: Naive Bayes results

Gradient Boosting classifier

Gradient Boosting Classifier (GBC) is a type of ensemble machine learning algorithm that uses multiple weak learners, such as decision trees, and combines their predictions to produce a stronger model. The GBC algorithm builds trees in a sequential manner, with each subsequent tree learning from the errors of the previous one. The final model is a weighted sum of the predictions of all the trees.

In the provided code, the GBC algorithm is used to train a classification model on a given dataset after preprocessing it. The preprocessed data is represented as TF-IDF vectors. The trained model is then used to make predictions on a test dataset, and its performance is evaluated using metrics such as accuracy, classification report, and confusion matrix. The accuracy score indicates the proportion of correctly classified instances, while the classification report and confusion matrix provide more detailed information about the model's precision, recall, and F1 score, and the number of true positives, true negatives, false positives, and false negatives, respectively.

Accuracy: 0.9946547884187082

F1 score 0.9943741209563994

```
Classification Report:
              precision    recall  f1-score   support
           0       1.00      0.99      0.99      4733
           1       0.99      1.00      0.99      4247
    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980

Confusion Matrix:
[[4690   43]
 [   5 4242]]
```

DistilBERTa

Moving on to the non-classical, transformer models, I chose to perform DistilBERTa on our data. DistilBERTa is a distilled version of the RoBERTa model that has been further compressed for efficiency and speed. Introduced by the Hugging Face team, it uses a combination of knowledge distillation and pruning to reduce the number of parameters without significantly sacrificing performance on various NLP tasks. With only 82 million parameters, DistilBERTa is significantly smaller and faster than RoBERTa, making it a great option for applications where computational resources are limited. It has been shown to achieve state-of-the-art performance on several NLP benchmarks, including text classification, question-answering, and sentiment analysis.

To begin with, the code accepts the input data and splits it into training, validation, and testing sets. Within the code we have functions like parameter_count() and metric_evaluation() which return the number of trainable parameters and calculate accuracy of the predicted values respectively.

We also have two very important functions: training and validate. Training function trains the model on a batch of data and calculates the loss and accuracy of that batch. These results are stored to be plotted later for visual understanding. The validate function evaluates the model on a batch of data and calculates the loss and accuracy also to be plotted later.

We initialize Distilbert using DistillbertForSequenceClassification. After experimenting with different parameters, we achieved the best results using epochs = 3, max_len = 256, batch_size = 32, and learning rate = 0.00001. These parameters yielded around 99% accuracy which is really good. Below we can see the results for other parameter values and later the accuracy and loss values for both training and validate functions.

Epoch	Batch_size	Learning_rate	Max_len	Training_accuracy	Validation_accuracy
3	32	0.00001	256	99.91	99.87
2	32	0.01	256	53.47	53.45

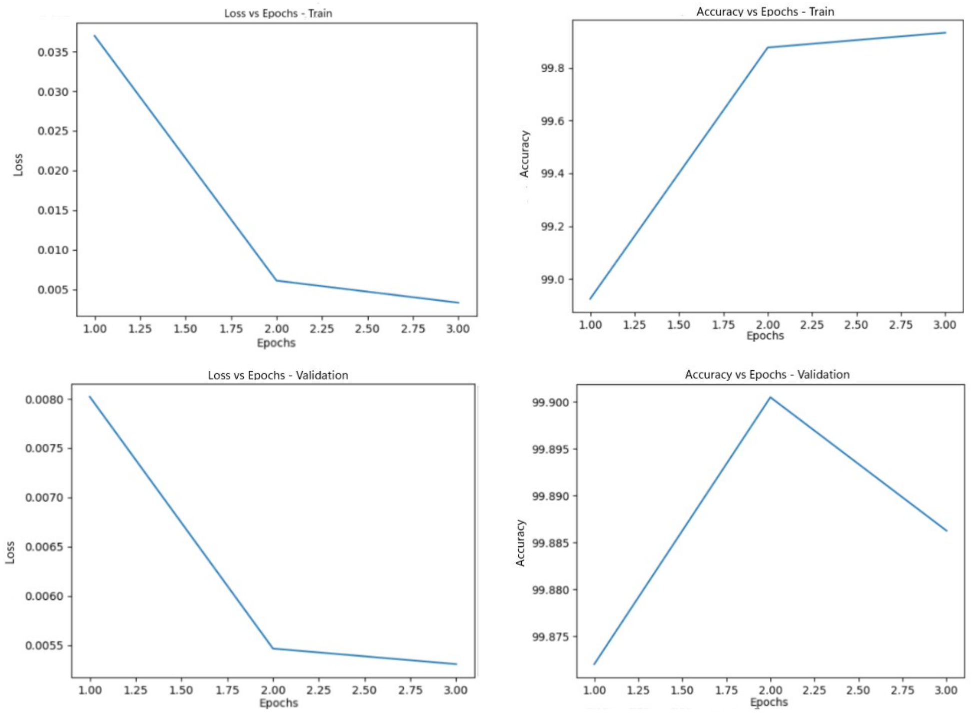


Figure 8 & 9: DistilBERTa results

RoBERTa

RoBERTa is a type of language model developed by Facebook AI Research (FAIR). It stands for "Robustly Optimized BERT approach", and is based on the BERT (Bidirectional Encoder Representations from Transformers) architecture. RoBERTa is designed to improve upon BERT's performance by using larger amounts of training data, longer training times, and tuning the hyperparameters more effectively.

Like BERT, RoBERTa is a transformer-based model that can be fine-tuned on a wide range of natural language processing (NLP) tasks, including text classification, question answering, and natural language generation. RoBERTa has achieved state-of-the-art performance on a number of benchmark NLP datasets, and is widely used in both academic research and industry applications.

Epoch	Batch_size	Learning_rate	Max_len	Training_accuracy	Validation_accuracy
3	32	0.00001	256	99.87	99.89
3	16	0.00001	256	99.94	99.9

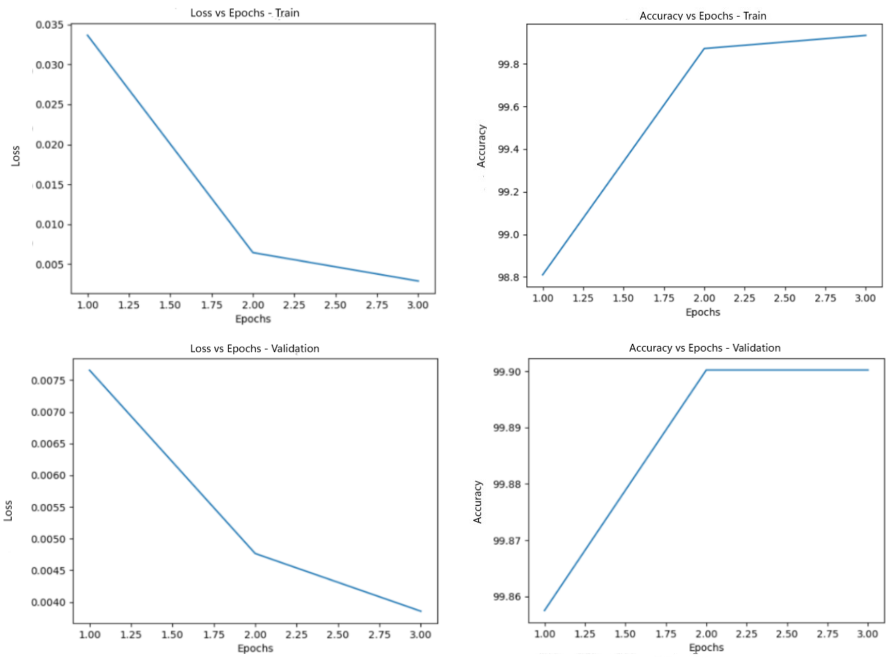


Figure 8 & 9: RoBERTa results

Conclusion

The classical models did not perform as bad as we expected them to. The best model with the highest accuracy in the above five models is the RoBERTa. However, if we are ready to sacrifice some of the prediction accuracy, DistilBERTa is towards a lighter processing load. It is also the most preferred model in the entire project for its balance between performance and speed.

In conclusion, the given data was sufficient and we are able to classify real and fake data with the help of several machine learning models.

References

1. https://www.analyticsvidhya.com/blog/2022/11/introduction-to-distilbert-in-student-model/#:~:text=DistilBERT%20model%20is%20a%20distilled,abilities%20and%20being%2060%25%20faster.
2. https://github.com/rasbt/stat453-deep-learning-ss21/blob/main/L19/distilbert-classifier/02_distilbert-with-scheduler.ipynb
3. https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset/code
4. https://chat.openai.com/
5. https://www.youtube.com/watch?v=emDmznRlsWw&ab_channel=SebastianRaschka
6. https://huggingface.co/distilbert-base-uncased
7. https://github.com/rasbt/stat453-deep-learning-ss21/blob/main/L19/distilbert-classifier/02_distilbert-with-scheduler.ipynb.
8. https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset
9. https://github.com/pytorch/fairseq/tree/master/examples/roberta
10. https://huggingface.co/transformers/
11. https://arxiv.org/abs/1907.11692

Amount of Code referenced

About 60% of my python code is referred from the internet and modified.