Final Term Project
DATS-6312 NLP for Data Science

Fake and Real News Dataset

Prof. Amir Jafari

Submitted by
Kohisha Aruganti

Introduction:

In today's world, staying informed about current events is more important than ever. However, with the rise of social media and the ease of spreading information online, it has become increasingly difficult to distinguish between real and fake news. This is a problem because fake news can spread quickly and cause harm by misinforming people about important issues.

In this current project we have decided to tackle this problem of classifying news articles as "real" or "fake". Our goal is to provide an NLP algorithm that can help determine whether the news articles are accurate or false. By doing so, we hope to improve the overall quality of news consumption and reduce the harmful effects of fake news.

Dataset Overview:

- The dataset is gathered from kaggle
- The dataset is classified into CSV files: fake and real news articles. The “True.csv” articles are sourced from reliable sources such as Reuters, the New York Times, and the BBC and “Fake.csv” articles are collected from unverified websites and wikipedia.
- Most of the articles are based on politics in the US.
- There are total of 44,919 observations of which 21,417 are True and 23,502 are Fake
- The objective of the project is to differentiate the “real” and “fake” news articles.

Data Pre-processing:

1. Removing URL’s:

```
def urls(text):
    return re.sub(r'http\S+', '', text)
```

This function called "urls" that takes a string of text as input. The purpose of this function is to remove any URLs (i.e., web links) from the text.

2. Applying lower casing

```
def to_lower(text):
    return text.lower()
```

This function called "to_lower" that takes a string of text as input. The purpose of this function is to convert all the characters in the input text to lowercase.

3. Removing Contractions

```
def remove_contractions(text):
    return ' '.join([contractions.fix(word) for word in text.split()])
```

The function called "remove_contractionsa" takes a string of text as input. The purpose of this function is to replace any contractions in the input text with their expanded forms. The function splits the input text into individual words using the "split()" function, then uses a list comprehension to iterate over each word and apply the "contractions.fix()" function to replace any contractions.

4. Removing Punctuation

```
def remove_punctuation(text):
  return text.translate(str.maketrans('', '', string.punctuation))
```

The function called "remove_punctuation" that takes a string of text as input. The purpose of this function is to remove any punctuation marks (such as periods, commas, and quotation marks) from the input text.

The "translate()" function takes two arguments: a translation table and a set of characters to delete. In this case, we're using the "str.maketrans()" function to create a translation table that maps each punctuation character to None (i.e., it deletes the character from the text). We're then passing this translation table to the "translate()" function along with the "string.punctuation" constant to delete all punctuation characters from the text.

5. Removing characters

```
def remove_characters(text):
    return re.sub('[^a-zA-Z]', ' ', text)
```

The remove_characters function takes a string text as input and removes any characters that are not alphabets (both uppercase and lowercase) using the re.sub() method in the re module.

6. Removing Stopwords

```
def remove_stopwords(text):
    stop_words = stopwords.words('english')
    return ' '.join([word for word in nltk.word_tokenize(text) if word not in stop_words])
```

The remove_stopwords function takes a text input and removes all the stopwords from it. Stopwords are commonly used words in a language that do not carry much meaning, such as "the", "a", "and", "of", etc.

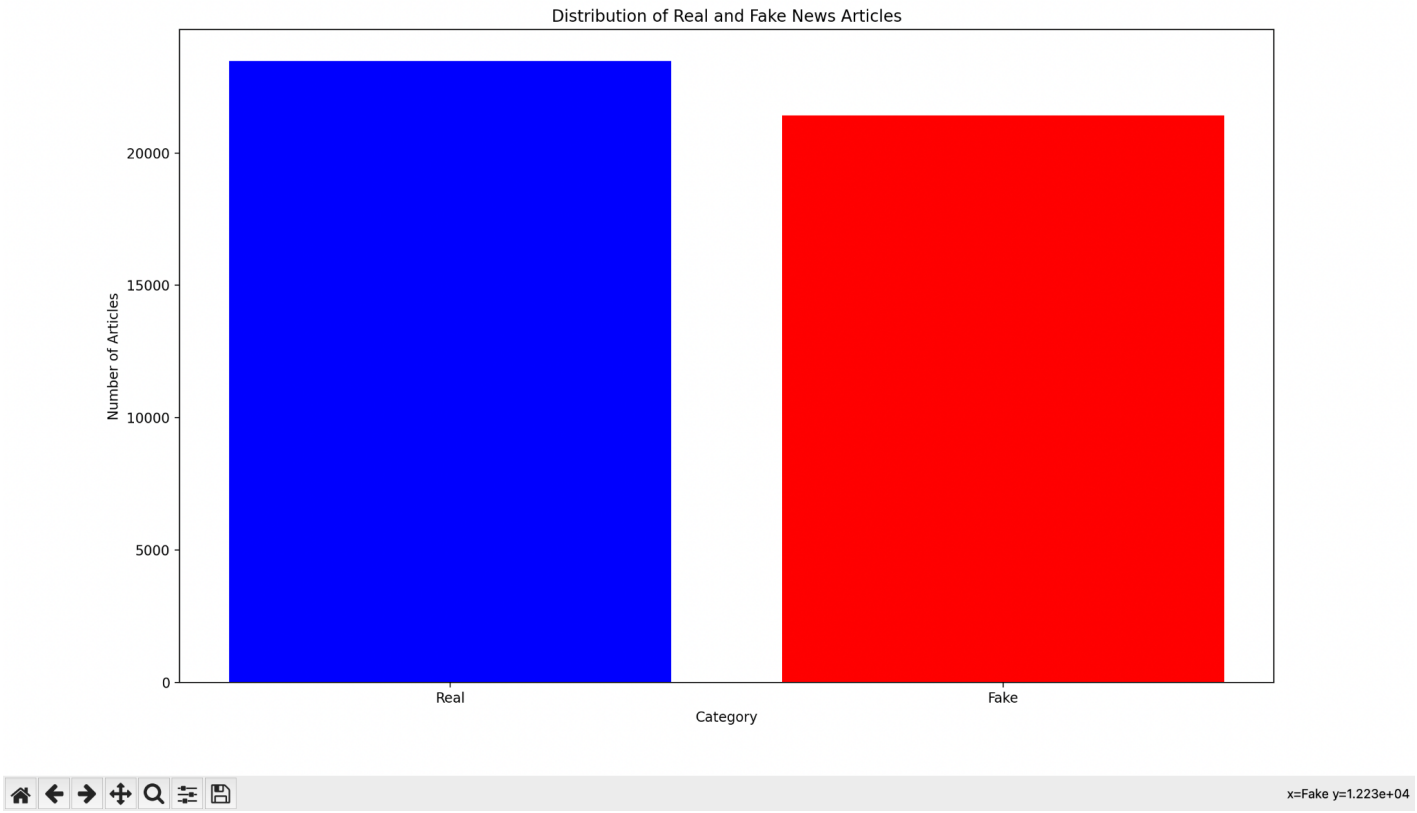
7.Lemmatize

```
def lemmatize(text):
    lem = WordNetLemmatizer()
    return ' '.join(lem.lemmatize(word) for word in text.split())
```

This function takes in a string of text and uses the WordNetLemmatizer from the NLTK library to lemmatize the words in the text. Lemmatization is the process of reducing a word to its base or root form

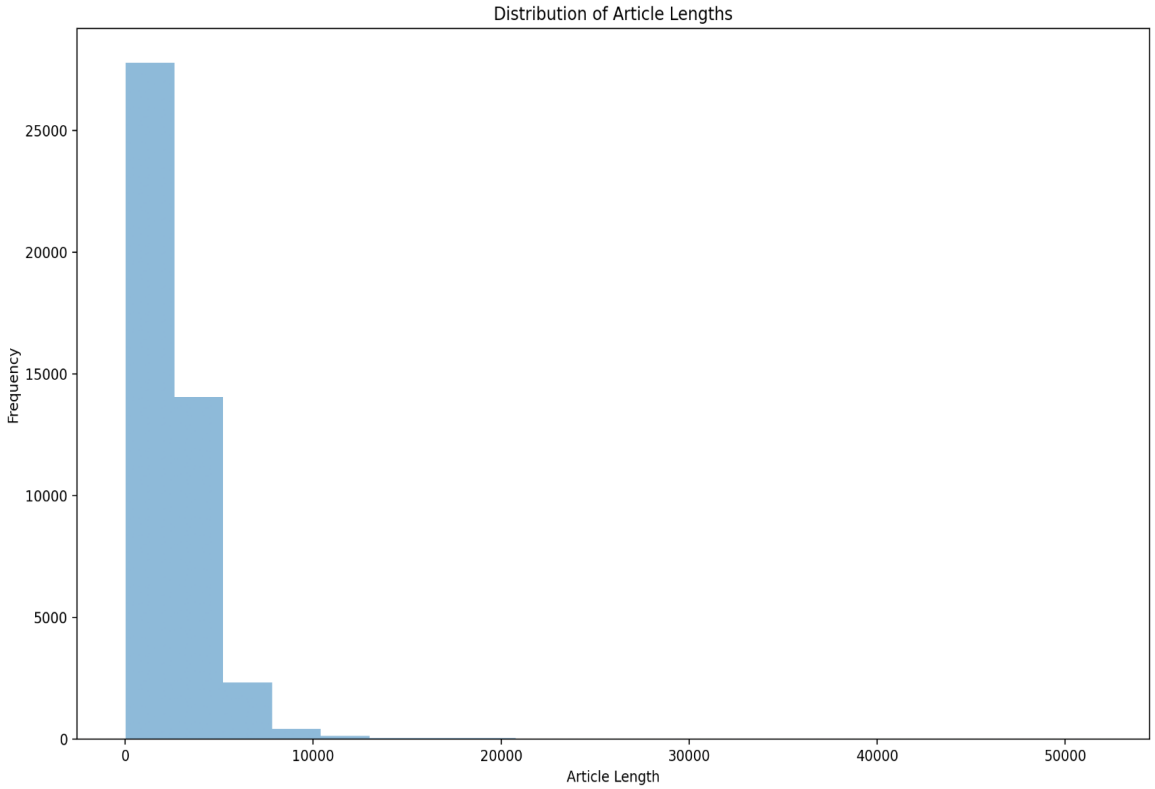
Exploratory Data Analysis:

Create a bar chart to visualize the distribution of fake and real news articles



From the bar graph, we can see that the distribution of real and fake articles is pretty balanced. Balanced dataset can help to ensure that the model is more accurate, reliable, and generalizable, while also reducing the risk of overfitting and improving interpretability.

Create a histogram to visualize the distribution of article lengths

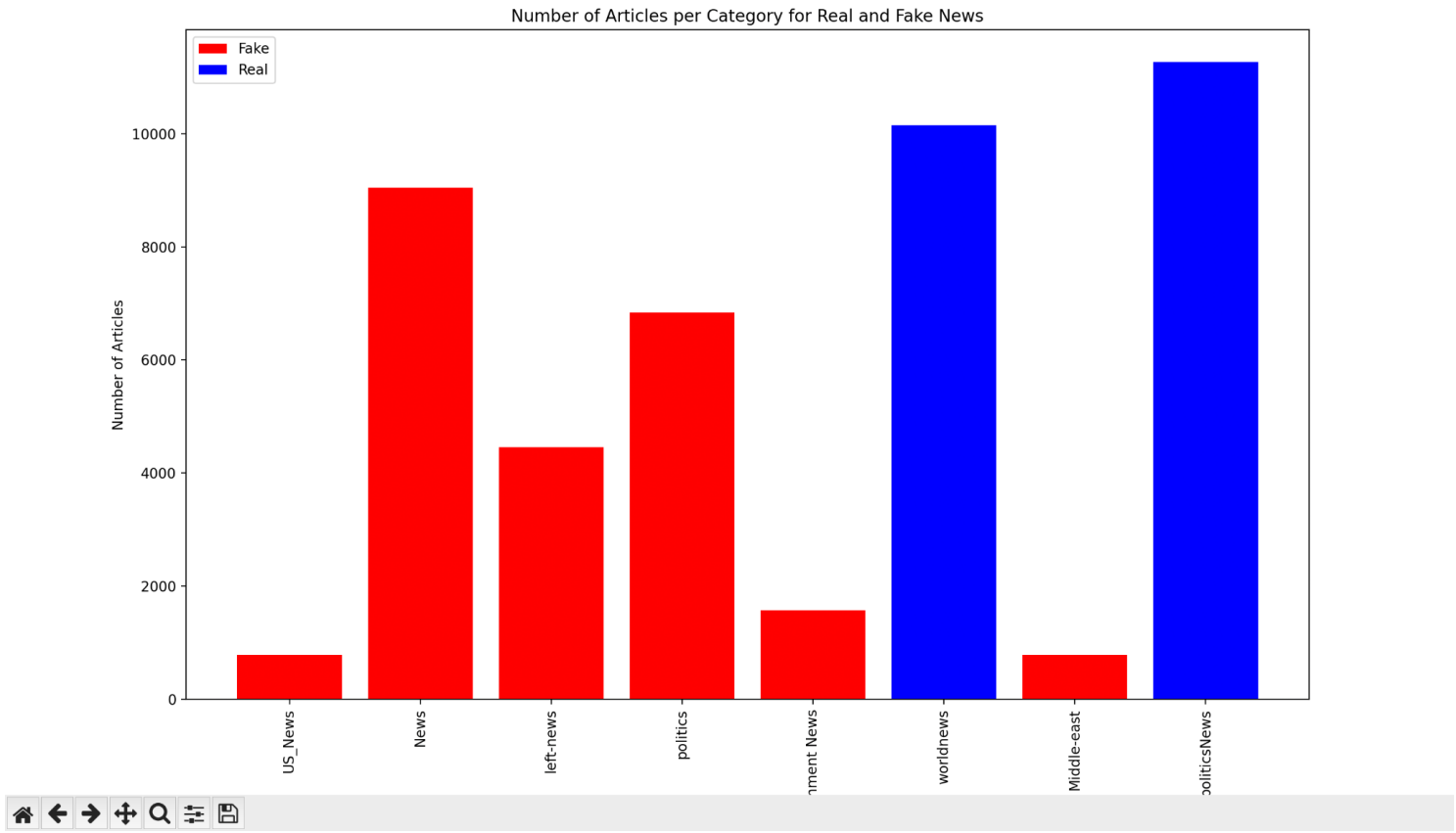


.

Histogram visualizes the distribution of article lengths in the dataset. Based on the histogram, we can see that the majority of articles in the dataset have a length between 0 and 10,000 words, with a peak around 2,500 frequency. There are also a smaller number of articles with lengths greater than 10,000 words, with a few outliers extending up to around 50,000.

Overall, the histogram suggests that the article lengths in the dataset follow a roughly normal distribution, with a few longer articles that are outliers. This information could be useful for understanding the characteristics of the dataset and potentially for designing algorithms that are optimized for processing articles of different lengths.

Create a stacked bar chart to visualize the distribution of articles by category for fake and real news



This stacked bar chart visualizes the distribution of articles by category for both fake and real news. It first counts the number of articles in each category for fake and real news separately, and then combines them into a single list of categories. It then creates a bar chart for each category, one for fake news and one for real news, and stacks them on top of each other.

Models:

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:

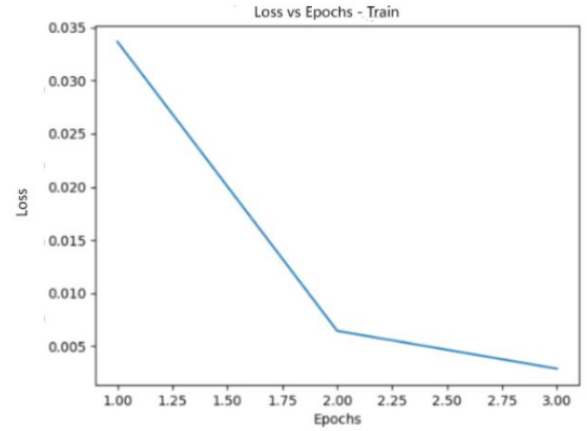
	precision	recall	f1-score	support
0	1.00	0.99	0.99	4733
1	0.99	1.00	0.99	4247
accuracy			0.99	8980
macro avg	0.99	0.99	0.99	8980
weighted avg	0.99	0.99	0.99	8980

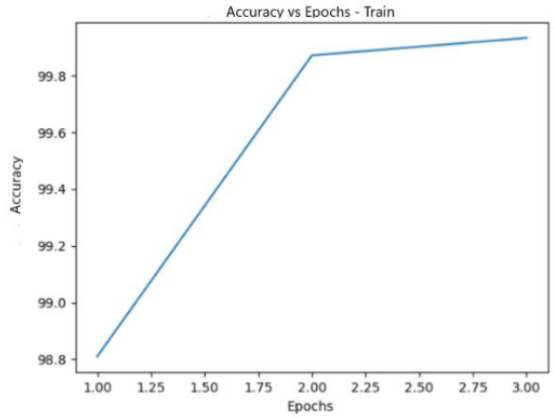
Confusion Matrix:
[[4690 43]
[5 4242]]

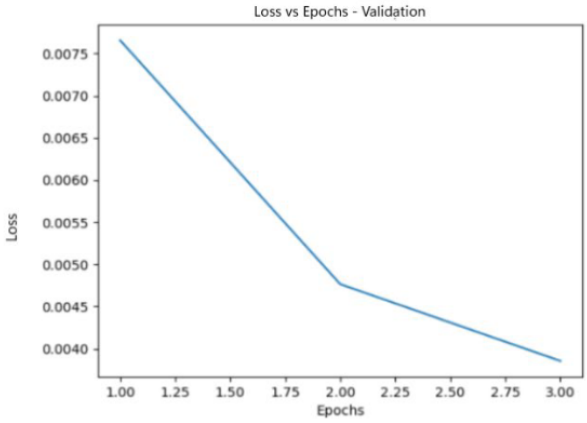
RoBERTa:

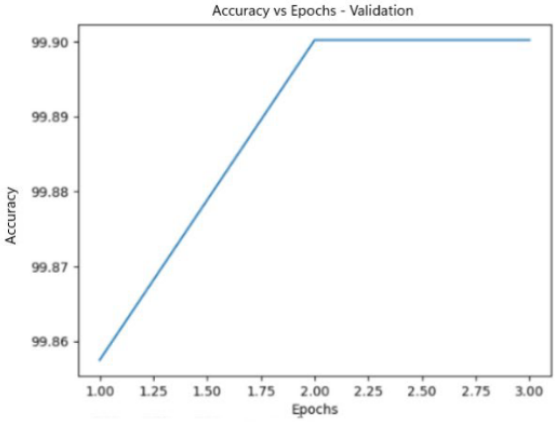
RoBERTa is a type of language model developed by Facebook AI Research (FAIR). It stands for "Robustly Optimized BERT approach", and is based on the BERT (Bidirectional Encoder Representations from Transformers) architecture. RoBERTa is designed to improve upon BERT's performance by using larger amounts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

Summary:

To Summarize,

- Gradient Boosting Classifier model has an f1 score of 0.9942115812917595
- RoBERTa model has an f1 score of “ 0.9996317991631799”.

Percentage of Code Written:

- The information about codes were referred from online. Approximately 45% of the code was referred from the internet.

References:

https://github.com/rasbt/stat453-deep-learning-ss21/blob/main/L19/distilbert-classifier/02_distilbert-with-scheduler.ipynb.
https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset
https://github.com/pytorch/fairseq/tree/master/examples/roberta
https://huggingface.co/transformers/
https://arxiv.org/abs/1907.11692