Palash S Patel | Sem 7 BDA | 22162121012

CGC Mini Project

Fake News Detection using Transformer

[Colab Link (Training): GullibleTransformer-1.ipynb]
[Github (Entire Project): github.com]

Dataset Link: kaggle.com

1. Problem statement:

The goal of this project is to develop a fake news detection system that classifies news headlines as real or fake using an encoder-only transformer model. Using the Kaggle Fake News Dataset, the system focuses exclusively on headline-level analysis to identify linguistic patterns and cues indicative of misinformation, ultimately improving the accuracy and efficiency of automated fake news classification.

2. Dataset Description

Dataset Name: Fake News Detection:
Key Characteristics:

1. Two Separate CSV Files:
   - Fake.csv: Contains news articles labeled as fake.
   - True.csv: Contains real (true) news articles.

2. Columns / Features:
   Each news item in both files has the following attributes:
   - title — The headline of the article.
   - text — The main body / content of the article.
   - subject — Topic or category of the news (e.g., “News”).
   - date — Publication date of the article.

3. Size / Volume

The dataset is fairly large (combined size ~43 MB as per Kaggle).
The exact number of articles is around 44,898 records (both fake + real).

4. Source / Provenance

The dataset is collected from real-world sources. The “truthful” articles come from credible / verified news outlets, while the fake ones are drawn from sources of misinformation.
Because it’s been used in published research (e.g., PLOS ONE), it has some academic validation.

3. Data Preprocessing

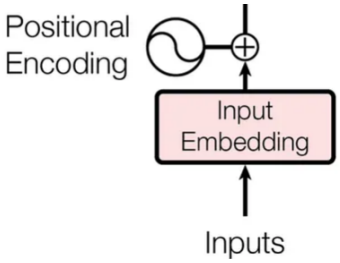
1. **Build the Tokenizer:** A large portion of the Dataset and its multiple columns are merged into a large corpus. This corpus (corpus file) is then used to create a Tokenizer.
   The Tokenizer of choice was my own implementation of the *BPE Tokenizer* (tokenizer code). The Tokenizer is then Trained and saved to file.

2. **Tokenize:** The tokenizer is then used to encode the actual dataset - (the title column).

3. **Transform into Tensors:** The tokens and their labels(0 - real ; 1 - fake) are stored into a torch.Tensor.

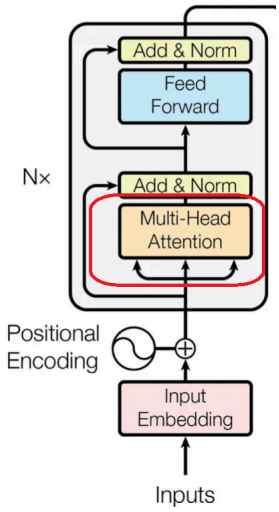
4. Model Building

- Embedding Layer (github.com):

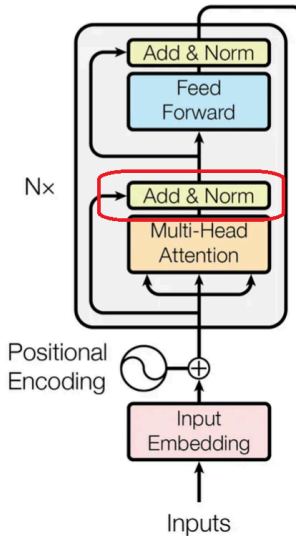


1. **Token Embedding Layer**: *Simple torch.nn.Embedding Layer*
2. **Positional Encoding Layer**: *Custom SinCosine Absolute Positional Encoding Layer. This precalculates the positions of tokens.*

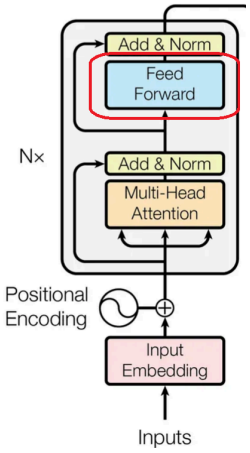
- Encoder Only Transformer Model (github.com):
    1. **Multihead Attention:** *Create K, Q, V from the input tensor. Partitions Q, K, V tensor into total - `n_heads` parts. Pass them through a `torch.nn.functional.scaled_dot_product_attention. Then applies a simple linear layer, to project output back to the same dimension as input and applies a dropout.*



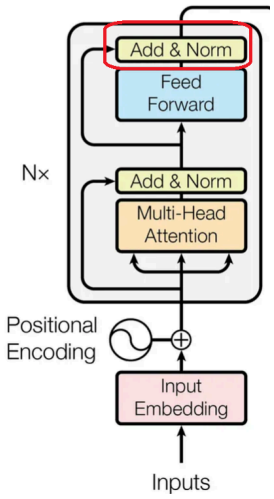
    2. **Residual Add + Normalizing Layer:** *Add before_attention_X + after_attention_X and apply a normalization layer - `torch.nn.functional.layer_norm`*



3. **Feed Forward:** *A normal FCC layer with 2 Linear Layers + 1 Activation Layer + 1 Dropout Layer*



4. **Residual Add + Normalizing Layer:** *Add before_Linear_X + after_Linear_X and apply a normalization layer - `torch.nn.functional.layer_norm`*



- Training Hyperparameter:

1. Number of Heads (For Multi Head Attention): 12
2. Number of Encoding Layers (MLA + FCC) : 12
3. Token Embedding Dimension: 768
4. Dropout: 0.0
5. Learning Rate: 0.0006
6. Max Token Size (For Attention Layers): 500
7. Epochs: 1

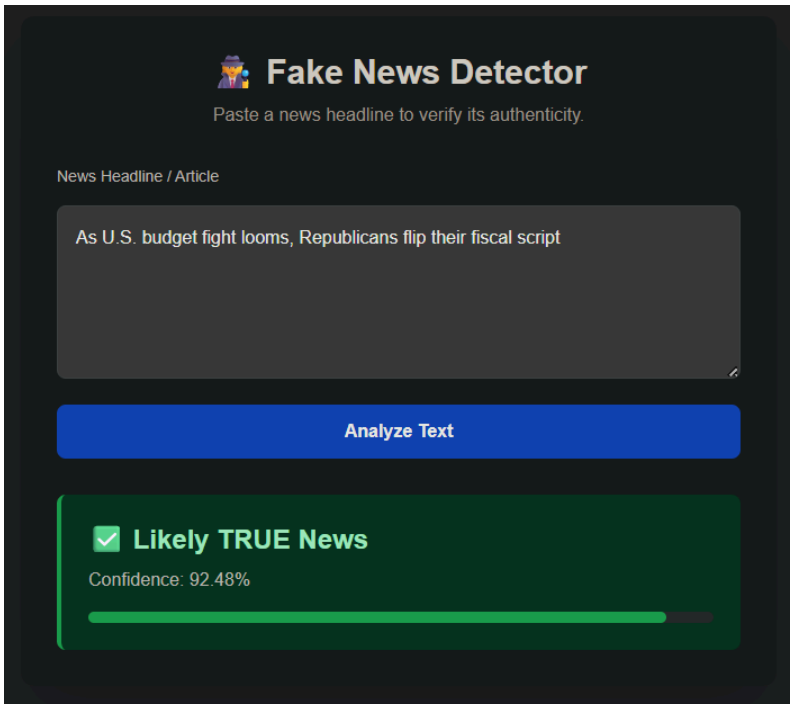
A single Epoch gets the job done - 96.89% Testing Accuracy, because dataset is large + the Batch Size is small - 12 (because of hardware constraints)

4. Training Results

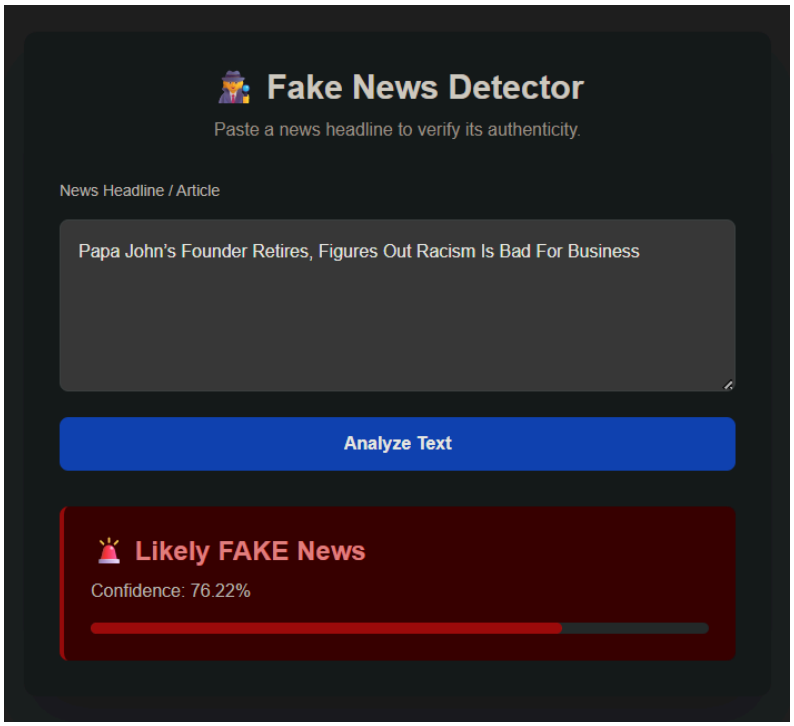
```
---------------  Testing  ----------------
✅ Saved Best Loss Model
✅ Saved Best Accuracy Model
Highest Testing Accuracy:  96.89309576837417  | epoch:  1
Least Testing Loss:  0.10045738680891587  | epoch:  1
Done !!
```

5. Model Output on Unseen Data

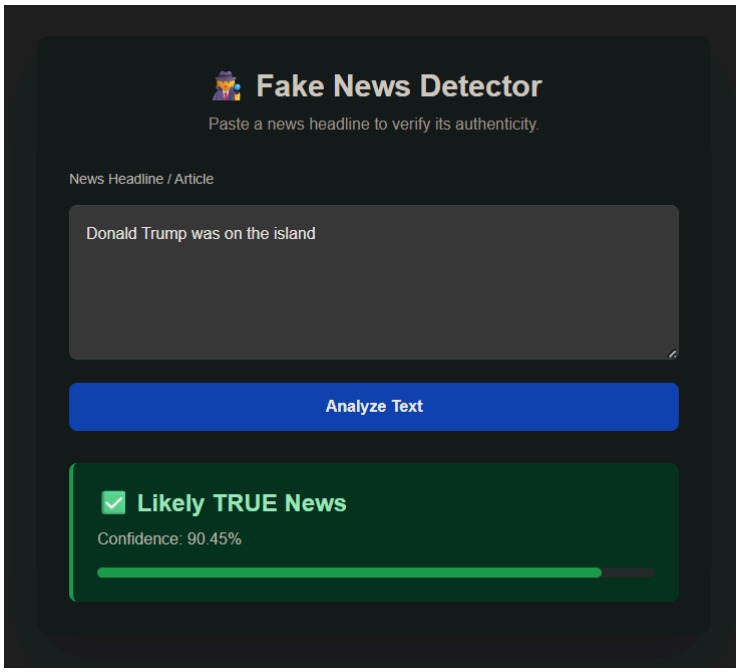
Example of True News:

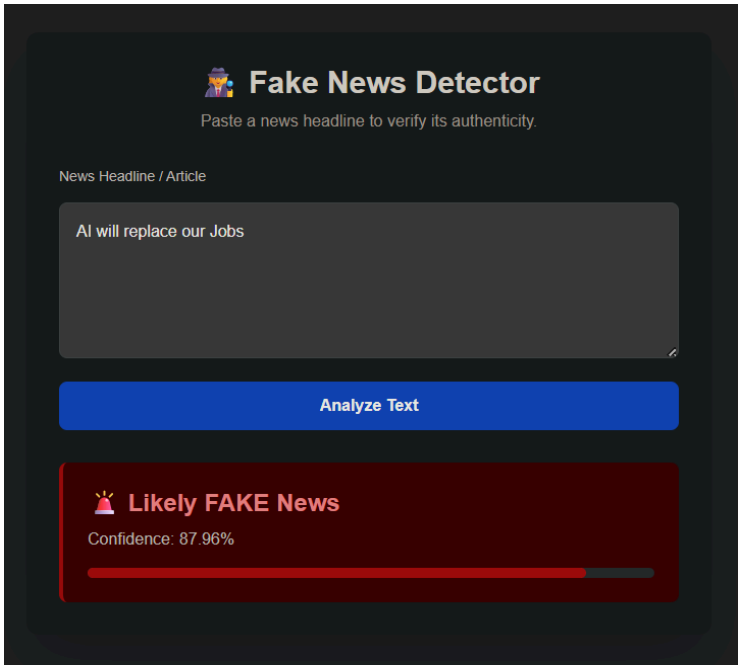


Example of Fake News:



→ Some other Examples 😁





6. Future Improvement

1. Convert the model to compare similarity between the input news and google search results of the new topic.
    - Show what differs, by providing multiple classes for classification such as `exaggerated`, `false`, `in correctly worded` and so on…
2. Make it multimodal, a lot of fake news use altered or AI generated images.
3. Find the exact words that have the highest weightage in a news being fake.