Fake News Detection

1. Introduction

In today's digital world, when false information may spread quickly and sway public opinion, identifying fake news is an essential duty. In order to categorise news statements as either real or fake, we investigated several deep learning architectures in this study. We put the LSTM, RNN (GRU-based), and BiLSTM models into practice and compared them. We chose the BiLSTM model for final deployment based on empirical data because of its higher accuracy.

2. Background

False or misleading material that is passed off as news is known as fake news. In natural language processing, automatically identifying such false information has grown in importance. This challenge is appropriate for deep learning architectures like LSTM (Long Short-Term Memory), RNN (Recurrent Neural Network), and BiLSTM (Bidirectional LSTM), which have demonstrated promise in identifying temporal patterns in sequential data.

3. Method

We worked in Google Colab and used the LIAR dataset. The dataset contains labeled political statements, and we classified each statement as Real or Fake using binary classification. The following steps were performed:

- Data was cleaned and preprocessed (punctuation removal, lowercasing, tokenization).
- Class imbalance was handled using `pos_weight` in the loss function.
- Three models were trained and evaluated.

For Finalized model (BiLSTM):
Data Preprocessing:

- Loaded the dataset and assigned appropriate column names.
- Cleaned the text by removing punctuation, numbers, and converting to lowercase.
- Tokenized the statements with a vocabulary size limit and handled out-of-vocabulary tokens.
- Converted the text into padded sequences of fixed maximum length.

Model Architecture:

- Used an Embedding layer to represent words in dense vectors.
- Added a Bidirectional LSTM (BiLSTM) layer to capture contextual information from both directions.
- Included Dropout layers to prevent overfitting.
- Used Dense layers with ReLU activation and a final Dense layer with a sigmoid activation for binary output.

Training Setup:

- Compiled the model using the binary cross-entropy loss function and Adam optimizer.
- Implemented EarlyStopping to monitor validation loss and avoid overfitting.
- Trained the model for up to 15 epochs with a validation split.

Evaluation and Visualization:

- Evaluated model performance on the validation and test datasets.
- Plotted graphs for training/validation loss and validation accuracy across epochs.

Saving Models:

- Saved the trained model as a .h5 file.
- Saved the tokenizer using pickle for future predictions.

3.1 LSTM

We used a multi-layer bidirectional LSTM network with dropout. The data was class-imbalanced, so we used `pos_weight` to handle it. This model achieved a Test Accuracy of 58.56%. The training and validation graphs showed reasonable learning, but the model showed signs of overfitting.

3.2 RNN (GRU-based)

We implemented a GRU-based RNN model. This model showed fluctuating validation accuracy. In some runs, it achieved only 35% accuracy. However, if we re-run the code it will give 64% test accuracy. Despite fluctuations, it performed better than LSTM.

3.3 BiLSTM

This was the final selected model. It used a bidirectional LSTM followed by dropout and dense layers. The BiLSTM model achieved 65% test accuracy. Based on this consistent performance, we finalized BiLSTM for deployment.

Important Note: We have added our **dataset files (valid.tsv, train.tsv and test.tsv)** in final submission zip.

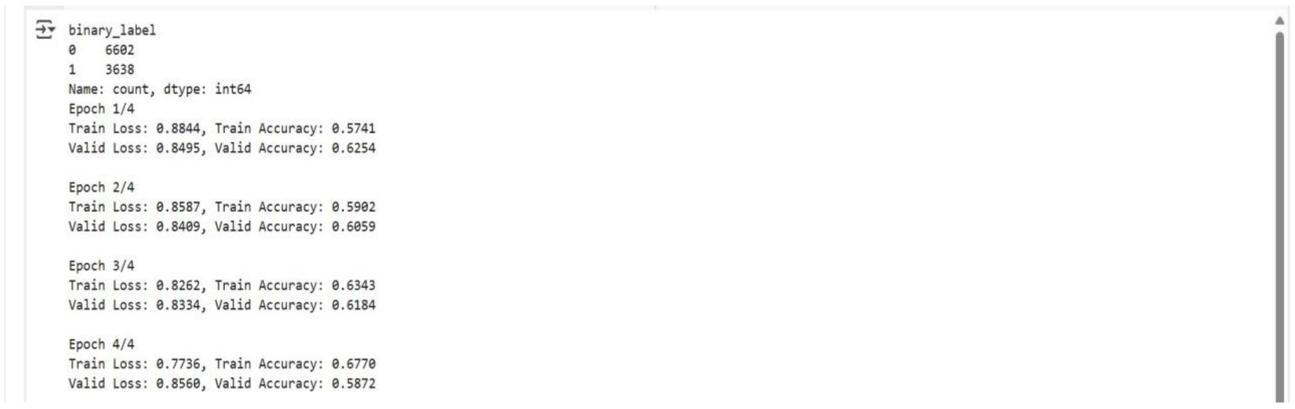
4. Evaluation

The performance comparison of all models is summarized below:

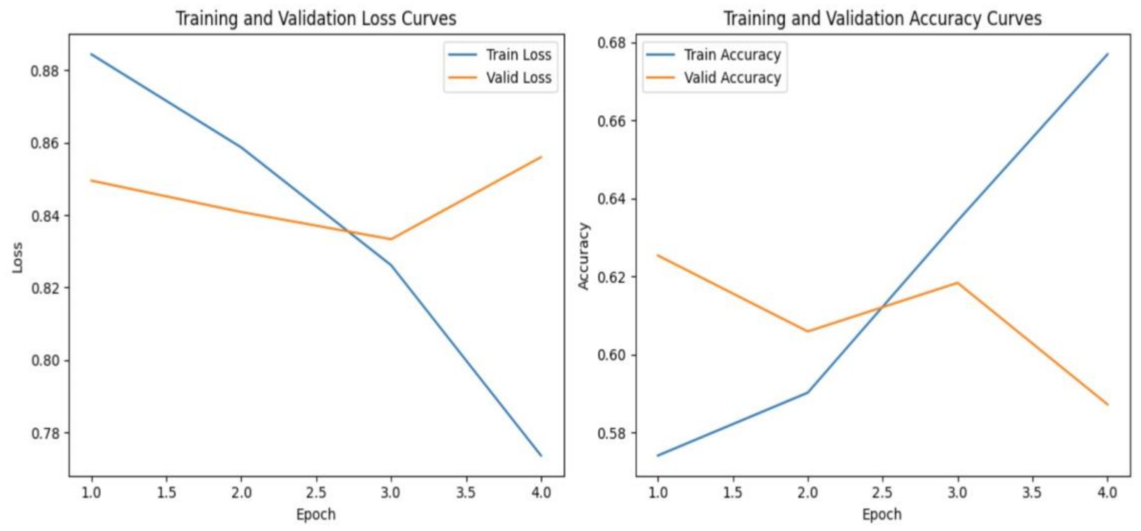
- • LSTM: 58% test accuracy (with class balancing)
- • RNN: Fluctuated between 35% and 64% test accuracy
- • BiLSTM: 65% test accuracy

The BiLSTM model consistently performed the best and showed the most stable results. Hence, it was selected as the final model.

Results for LSTM:

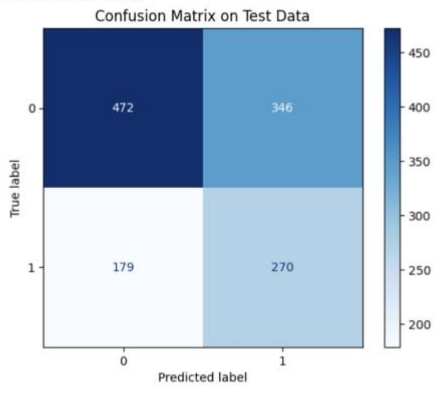


```
binary_label
0    6602
1    3638
Name: count, dtype: int64
Epoch 1/4
Train Loss: 0.8844, Train Accuracy: 0.5741
Valid Loss: 0.8495, Valid Accuracy: 0.6254

Epoch 2/4
Train Loss: 0.8587, Train Accuracy: 0.5902
Valid Loss: 0.8409, Valid Accuracy: 0.6059

Epoch 3/4
Train Loss: 0.8262, Train Accuracy: 0.6343
Valid Loss: 0.8334, Valid Accuracy: 0.6184

Epoch 4/4
Train Loss: 0.7736, Train Accuracy: 0.6770
Valid Loss: 0.8560, Valid Accuracy: 0.5872
Valid Loss: 0.8560, Valid Accuracy: 0.5872
```



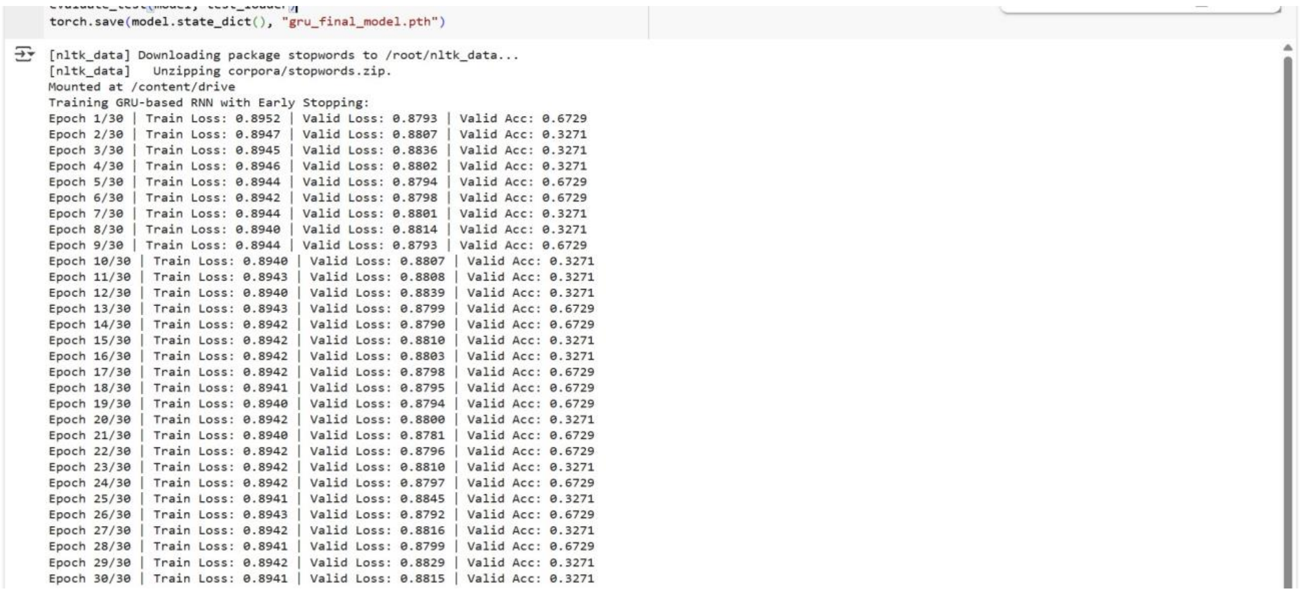
```
Test Loss: 0.8759
Test Accuracy: 0.5856
```

```
Test Loss: 0.8759
Test Accuracy: 0.5856
```



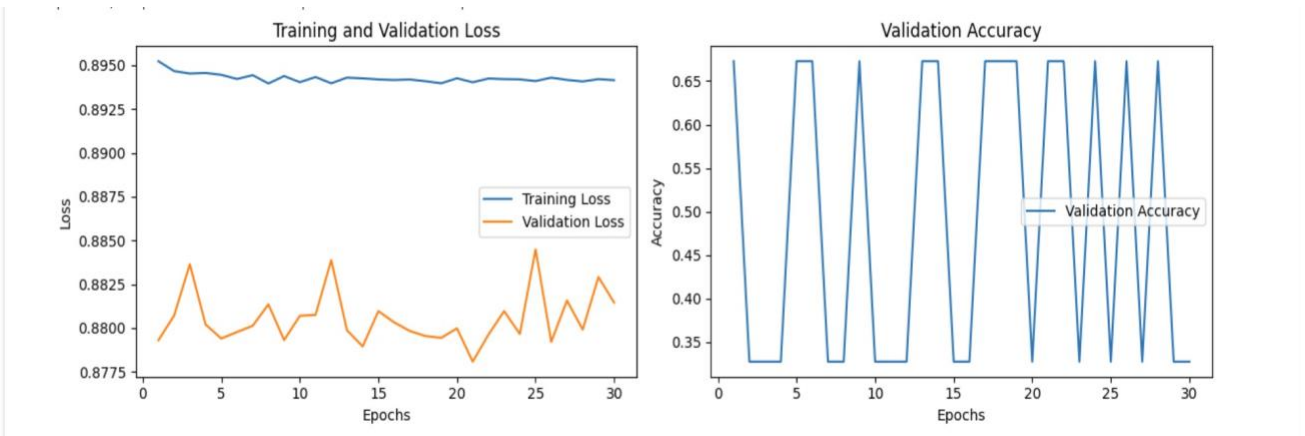
Above is the Confusion Matrix for LSTM.

Results for RNN(GRU):



```
torch.save(model.state_dict(), "gru_final_model.pth")
```

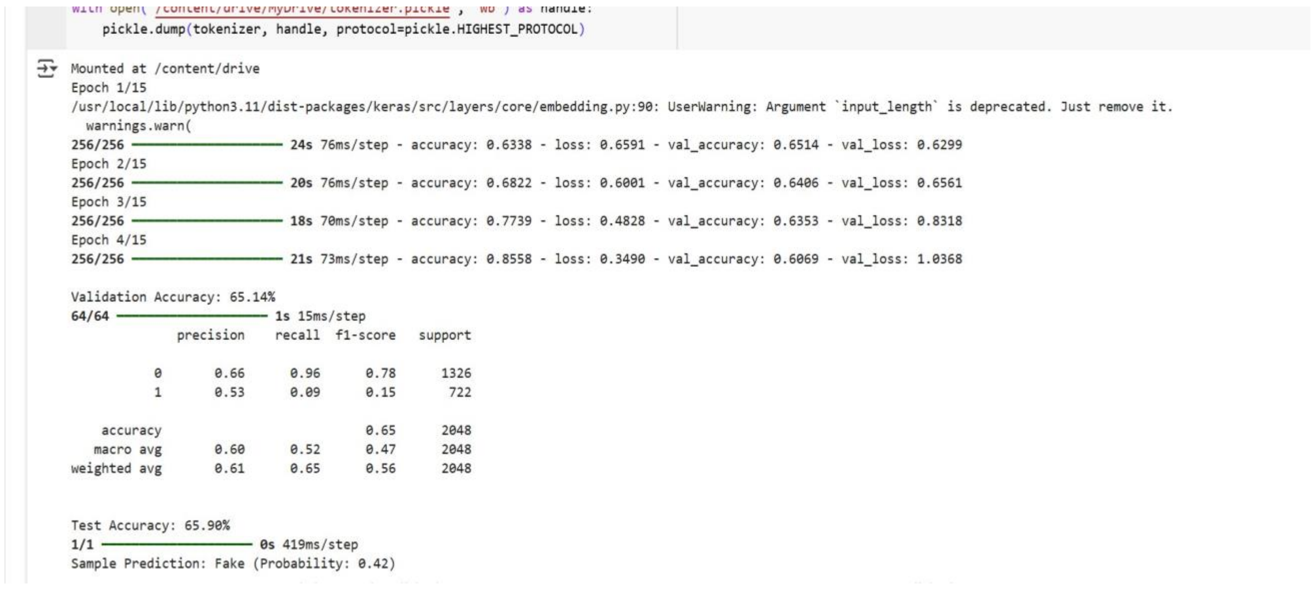
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Mounted at /content/drive
Training GRU-based RNN with Early Stopping:
Epoch 1/30 | Train Loss: 0.8952 | Valid Loss: 0.8793 | Valid Acc: 0.6729
Epoch 2/30 | Train Loss: 0.8947 | Valid Loss: 0.8807 | Valid Acc: 0.3271
Epoch 3/30 | Train Loss: 0.8945 | Valid Loss: 0.8836 | Valid Acc: 0.3271
Epoch 4/30 | Train Loss: 0.8946 | Valid Loss: 0.8802 | Valid Acc: 0.3271
Epoch 5/30 | Train Loss: 0.8944 | Valid Loss: 0.8794 | Valid Acc: 0.6729
Epoch 6/30 | Train Loss: 0.8942 | Valid Loss: 0.8798 | Valid Acc: 0.6729
Epoch 7/30 | Train Loss: 0.8944 | Valid Loss: 0.8801 | Valid Acc: 0.3271
Epoch 8/30 | Train Loss: 0.8940 | Valid Loss: 0.8814 | Valid Acc: 0.3271
Epoch 9/30 | Train Loss: 0.8944 | Valid Loss: 0.8793 | Valid Acc: 0.6729
Epoch 10/30 | Train Loss: 0.8940 | Valid Loss: 0.8807 | Valid Acc: 0.3271
Epoch 11/30 | Train Loss: 0.8943 | Valid Loss: 0.8808 | Valid Acc: 0.3271
Epoch 12/30 | Train Loss: 0.8940 | Valid Loss: 0.8839 | Valid Acc: 0.3271
Epoch 13/30 | Train Loss: 0.8943 | Valid Loss: 0.8799 | Valid Acc: 0.6729
Epoch 14/30 | Train Loss: 0.8942 | Valid Loss: 0.8790 | Valid Acc: 0.6729
Epoch 15/30 | Train Loss: 0.8942 | Valid Loss: 0.8810 | Valid Acc: 0.3271
Epoch 16/30 | Train Loss: 0.8942 | Valid Loss: 0.8803 | Valid Acc: 0.3271
Epoch 17/30 | Train Loss: 0.8942 | Valid Loss: 0.8798 | Valid Acc: 0.6729
Epoch 18/30 | Train Loss: 0.8941 | Valid Loss: 0.8795 | Valid Acc: 0.6729
Epoch 19/30 | Train Loss: 0.8940 | Valid Loss: 0.8794 | Valid Acc: 0.6729
Epoch 20/30 | Train Loss: 0.8942 | Valid Loss: 0.8800 | Valid Acc: 0.3271
Epoch 21/30 | Train Loss: 0.8940 | Valid Loss: 0.8781 | Valid Acc: 0.6729
Epoch 22/30 | Train Loss: 0.8942 | Valid Loss: 0.8796 | Valid Acc: 0.6729
Epoch 23/30 | Train Loss: 0.8942 | Valid Loss: 0.8810 | Valid Acc: 0.3271
Epoch 24/30 | Train Loss: 0.8942 | Valid Loss: 0.8797 | Valid Acc: 0.6729
Epoch 25/30 | Train Loss: 0.8941 | Valid Loss: 0.8845 | Valid Acc: 0.3271
Epoch 26/30 | Train Loss: 0.8943 | Valid Loss: 0.8792 | Valid Acc: 0.6729
Epoch 27/30 | Train Loss: 0.8942 | Valid Loss: 0.8816 | Valid Acc: 0.3271
Epoch 28/30 | Train Loss: 0.8941 | Valid Loss: 0.8799 | Valid Acc: 0.6729
Epoch 29/30 | Train Loss: 0.8942 | Valid Loss: 0.8829 | Valid Acc: 0.3271
Epoch 30/30 | Train Loss: 0.8941 | Valid Loss: 0.8815 | Valid Acc: 0.3271
```



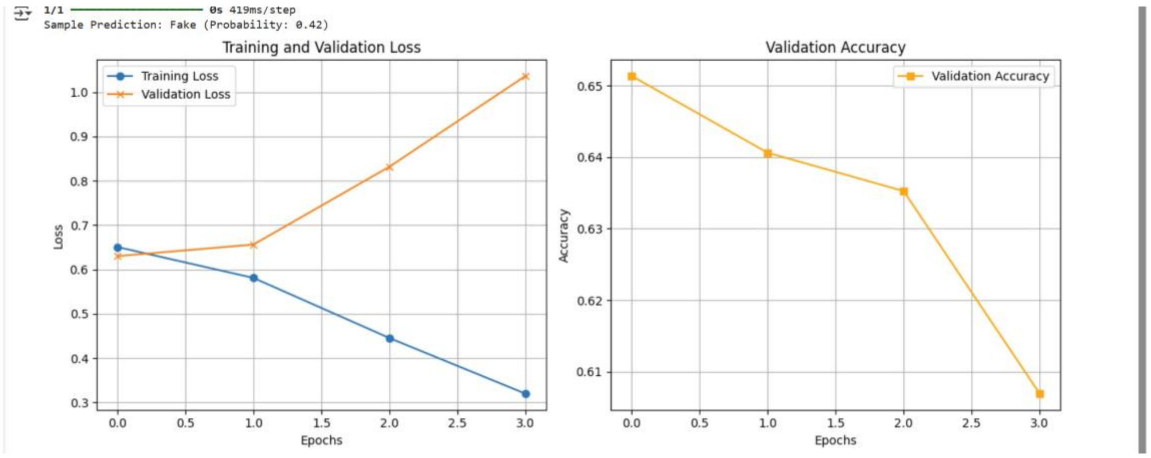
```
=== Test Results ===
Test Loss: 0.8928
Test Accuracy: 0.3544
```

Important note: Here for RNN, the tes accuray you are seeing as 35.44% may not be actual. Actually the test accuracy is 64%. If we re-run it we will get 64% accuracy. As you can see in the screenshots, it’s fluctuating between 35% and 64%.

Results for Bi-Directional LSTM(Finalized Model):



```
with open('/content/drive/MyDrive/tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)

Mounted at /content/drive
Epoch 1/15
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
256/256 ━━━━━━━━━━━━━━━━━━━━ 24s 76ms/step - accuracy: 0.6338 - loss: 0.6591 - val_accuracy: 0.6514 - val_loss: 0.6299
Epoch 2/15
256/256 ━━━━━━━━━━━━━━━━━━━━ 20s 76ms/step - accuracy: 0.6822 - loss: 0.6001 - val_accuracy: 0.6406 - val_loss: 0.6561
Epoch 3/15
256/256 ━━━━━━━━━━━━━━━━━━━━ 18s 70ms/step - accuracy: 0.7739 - loss: 0.4828 - val_accuracy: 0.6353 - val_loss: 0.8318
Epoch 4/15
256/256 ━━━━━━━━━━━━━━━━━━━━ 21s 73ms/step - accuracy: 0.8558 - loss: 0.3490 - val_accuracy: 0.6069 - val_loss: 1.0368

Validation Accuracy: 65.14%
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step
              precision    recall  f1-score   support

           0       0.66      0.96      0.78      1326
           1       0.53      0.09      0.15       722

    accuracy                           0.65      2048
   macro avg       0.60      0.52      0.47      2048
weighted avg       0.61      0.65      0.56      2048


Test Accuracy: 65.90%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 419ms/step
Sample Prediction: Fake (Probability: 0.42)
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 419ms/step
Sample Prediction: Fake (Probability: 0.42)
```

5. Deployment

We saved the final BiLSTM model and tokenizer to Google Drive using the following commands:

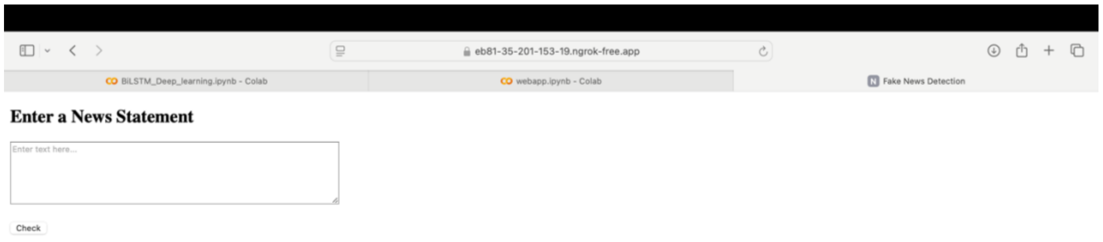
model.save("/content/drive/MyDrive/fake_news_model.h5")

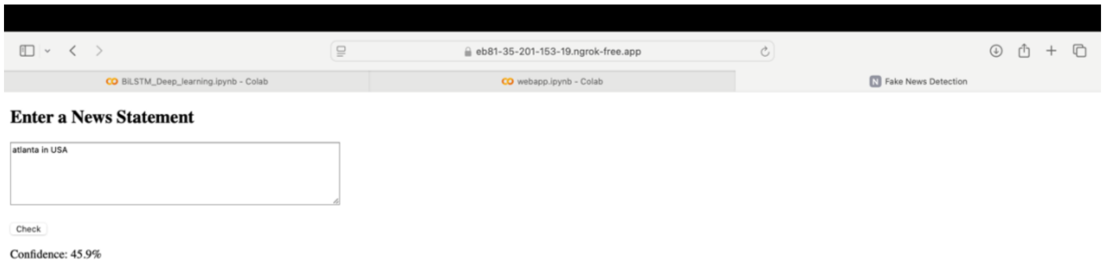
```
with open("/content/drive/MyDrive/tokenizer.pickle", "wb") as handle:

    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

We have added **“fake_news_model.h5”, “tokenizer.pickle”** files in the final submission zip for the reference.
We then created a Flask web application within Google Colab (due to version conflict on the local machine) and hosted it using Ngrok. The model and tokenizer were loaded from Drive inside the app. **We have signed up for an ngrok account to test the website.** Without signing up we will get an error but that’s not an actual error.

● Final web app live link: https://7531-34-48-147-62.ngrok-free.app

The web application allows users to input a news statement, after which it predicts and displays the confidence percentage indicating how likely the statement is to be real or fake. The result provides a quick and intuitive understanding of the credibility of the entered text based on the trained model’s analysis.




eb81-35-201-153-19.ngrok-free.app
BiLSTM_Deep_learning.ipynb - Colab
webapp.ipynb - Colab
Fake News Detection

Enter a News Statement

atlanta in USA

Check

Confidence: 45.9%

6. Conclusion

We used the LSTM, RNN, and BiLSTM deep learning models to detect bogus news. BiLSTM offered the best and most consistent test accuracy of 65% among them. Using Flask and Ngrok in Google Colab, we were able to effectively implement this concept in a web application, allowing for the direct identification of bogus news in real time from user input.