Deep Learning (COSC 2779/2972) – Assignment 2 – 2023
<Harshith Hullakere Siddegowda>(<s3914263>)

Problem Definition and Background

The project aims to develop a end to end deep learning model for stance classification in tweets, focusing on five politically-charged target topics: "Atheism," "the Feminist Movement," "Climate Change is a Real Concern," "Legalization of Abortion," and "Hillary Clinton." The **primary goal** is to classify tweets into one of three categories: FAVOR, AGAINST, or NEITHER, indicating whether the tweet expresses support, opposition, or neither toward the specified target topic.

Challenges:

Ambiguity: Some tweets may not explicitly express support or opposition.

Context Dependency: Stance can be context-dependent and may vary across topics.

Data Imbalance: The dataset may have an uneven distribution of stances and topics.

Evaluation Framework

Macro F1 Score: The macro F1 score is an essential metric for stance classification. It considers precision and recall for each stance class independently and then takes the average. This metric gives equal importance to all classes, making it a suitable choice for multi-class classification tasks with imbalanced data.

Precision and Recall: These metrics are crucial for stance classification, particularly when dealing with imbalanced datasets. Precision measures the accuracy of the model's predictions, ensuring that when it predicts a stance, it is likely to be correct. Recall assesses the model's ability to capture all relevant instances of a stance, reducing false negatives. In politically-charged topics, balancing accuracy and completeness is vital, making these metrics indispensable.

(Weighted) F1 Score: This metric provides a balanced assessment of the model's performance while considering the practical importance of different stance classes, which is particularly crucial when dealing with imbalanced datasets.

Confusion Matrix: This visual representation of the model's performance offers in-depth insights. It categorizes predictions into true positives, true negatives, false positives, and false negatives for each stance class. The confusion matrix aids in identifying specific strengths and weaknesses, pinpointing where the model excels and where it struggles, and enabling class-wise evaluations.

Approach & Justifications

❖ **Exploratory Data Analysis (EDA):**

EDA phase,I have made several noteworthy observations: **Stance Distribution:** The dataset exhibits an imbalance, with "AGAINST" being the most prevalent stance (1395 instances), followed by "NONE" (766) and "FAVOR" (753).**Target Distribution:** The target topics are almost evenly distributed across all tweets, indicating a balanced representation of political instances.**Sentimental Analysis:** Sentiment labels, including negative (1762 instances), positive (963 instances), and other (189 instances), provide an insight into the overall sentiment of tweets in the dataset.**Target Distribution with Stance Categories:** Specific patterns emerge within different stance categories:

- "AGAINST" stance has minimal representation (less than 4%) related to the "Climate Change is a Real Concern" target.
- "FAVOR" stance has significantly fewer instances compared to the "AGAINST" stance, with a nearly 50% reduction.
- The "NONE" stance is roughly equally represented as the "FAVOR" stance, indicating that it has fewer instances compared to "AGAINST."

Word Cloud Analysis: For each political target stance, I have created word clouds to visualize word frequencies in tweets related to specific topics. Each word cloud was generated to capture the unique vocabulary associated with these topics.

Data Loader(Data Generator Class) and NLP Cleaning

- In the **Data Loader** and **NLP cleaning process**, I have created a versatile data generator class, known as DataGenerator, to facilitate the efficient loading and preprocessing of text data for our stance classification project. This class is designed to work seamlessly with TensorFlow and provides a systematic approach to handling the complexities of text data.

- Within the **DataGenerator class,** several critical functionalities have been implemented. First, the class is initialized with key parameters such as the data frame, batch size, number of classes, and data shuffling preferences. To facilitate the training process, the class processes the tweet, target, and stance information from the provided dataset, ensuring that it is ready for use in a deep learning model. Importantly, it encodes stance labels into one-hot vectors, making them suitable for multiclass classification.

- The class implements methods to determine the number of batches per epoch (__len__) and shuffles data indexes after each epoch (on_epoch_end). During batch generation (__getitem__), the class preprocesses text data, converts it into TensorFlow tensors, and handles the necessary encoding of stance labels. The **NLP text preprocessing** includes tasks such as cleaning text by removing extra whitespaces, emojis, URLs, and special characters. It also involves tokenization, converting text to lowercase, and eliminating punctuation and non-alphanumeric characters However, I Am not removing stop words because it loses the context of the sentence.

- Moreover, it's worth noting that when the DataGenerator generates **batches**, the shape of the data it provides is as follows: Tweet Shape: (32,), Target Shape: (32,), and Stance Shape: (32, 3).

❖ Building The Model:

Why I have selected the BERT models rather than other embeddings is, with their bidirectional contextual understanding, pre-trained knowledge, fine-tuning capabilities, and a range of available architectures, make them a compelling choice for NLP tasks, such as political stance classification in tweets. These models have proven successful in a wide array of NLP applications and continue to be at the forefront of natural language understanding technology.

- **Baseline Model Architecture**: The baseline classifier model: I am using a **Small BERT model** as a core component to experiment initially, which offers a balance between model size, speed, and quality. They are particularly suitable for real-world applications where efficient resource usage and fast inference times and they are essential capabilities. It features an architecture with **two input layers** for tweet and target text, followed by **preprocessing layers** to tokenize and preprocess the text data. The **BERT encoder** extracts contextual embeddings, and the concatenated outputs capture the relationship between the tweet and target text.Secondly, added **Dropout Layer** A dropout layer with a 0.1 dropout rate is included to reduce overfitting and improve generalization.added, **Classifier Layer** The model utilizes a dense layer with softmax activation for multi-class classification into "FAVOR," "AGAINST," or "NONE."added, **Regularization** L2 regularization with random value of 0.00001 is applied to the classifier layer to control overfitting, encouraging smaller weights for better generalization.

Problem encountered: After training for 25 epochs, the model reached a categorical accuracy of roughly 55.68%. It's important to note that this accuracy level stabilizes at around 55% starting from the 15th epoch. The validation accuracy follows a similar pattern, with a consistent value of about 56.65% and minimal fluctuations. Concerning the reduction in loss, it becomes marginal after a certain point during training.Secondly, Challenges observed include the model's inability to capture text nuances, emphasizing the need for improved textual understanding. I added learning rate but there was no significant change in my validation and training accuracy.

Transition from Baseline to Bidirectional LSTM with BERT

I am incorporating bidirectional **Long Short-Term Memory (LSTM) layers**, further enhancing the model's ability to capture contextual information and nuances in text. In addition to this, I leveraged the power of **Google's 'talkheads_ggelu_bert_en_base'** pre-trained BERT embeddings for tweet and target text processing.

- **Input Layer**: Accepts the tweet and target text as input. **Pre-processing Layer**: Utilizes TensorFlow Hub to apply the preprocessing transformations specific to the chosen BERT model. **BERT-Encoder Layers**: Incorporate the BERT encoder to extract contextual embeddings from both tweet and target text. The trainable parameter is set to 'False,' ensuring that the pre-trained BERT embeddings are not modified during training. **Concatenation Layer**: Concatenates the pooled output embeddings from tweet and target, forming a comprehensive representation that captures the relationship between the two. **Reshape Layer**: Reshapes the concatenated output, assuming a sequence length of 32 (adjustable as needed). **Bidirectional LSTM Layers**: Empower the model to capture temporal relationships and context. These layers are bidirectional to capture information from both directions. **Dropout Layer**: Helps prevent overfitting by randomly dropping out a fraction of units during training. **Stance Classification Layer**: The model's primary classification layer, which initially includes a dense layer with a ReLU activation function for feature extraction. **Additional Stance Classification Layer**: A second dense layer with L2 regularization further refines the model for stance classification. The softmax activation function is used to predict one of the three classes: 'FAVOR,' 'AGAINST,' or 'NONE.'

Problem encountered: After adding Bidirectional LSTM Layers and additional Dense Layer my training accuracy start increasing and reached (0.6181) but validation tend to decrease significantly after few epochs(0.5161), Its clearly an overfitting then I started doing my Hyperparameter tuning, mainly focusing on balancing my dataset, Learning rate Batch size and Regularization.

Experiments & Tuning

- **Batch Size Experiment:** In a quest for the optimal training batch size, a range of values including 8, 16, and 32 were experimented with. The **batch size of 32** emerged as the most effective, indicating that larger batch sizes enabled more efficient learning by updating model weights more frequently. It also promoted better generalization, enhancing the model's overall performance. **Number of Epochs Experiment:** The investigation extended to the number of training epochs, varying between 25, 40, 50, 70, and 100. To prevent overfitting and pinpoint the sweet spot for model accuracy, **early stopping** was introduced. Surprisingly, the best results were observed with 25 epochs, achieving a harmonious blend of performance and training efficiency.
- **Data Augmentation:** Addressing data imbalance challenges, techniques like Random Sampler and ADASYN Data Oversampler were explored. However, the **'bert-base-uncased' model** emerged as the most effective solution. It significantly balanced 'NONE' and 'FAVOR' stance classes, from 613 and 603 samples to 900 each, while considering the 'AGAINST' class with 1115 samples. The augmentation process not only addressed class imbalance but also preserved contextual relevance, significantly enhancing the model's ability to understand nuanced political language.
- **Optimizer Selection:** Three different optimizers - RMSprop, SGD, and Adam - were considered, each influencing model weight updates differently during training. The experiments covered **multiple learning rates** [0.01, 0.001, 0.0001], crucial for efficient convergence. **Adam and RMSprop** yielded the highest F1 scores (Macro) of 0.54, with Precision at 0.64 for the Stance output.
- **Effect of Lambda (Regularization Strength):** Experimenting with different lambda values (0.05, 0.01, 0.005, and 0.001) highlighted the critical role of regularization in curbing overfitting. A lambda value of 0.001 emerged as the optimal choice, striking a balance between preventing overfitting and maintaining robust generalization.

Ultimate Judgment, Analysis & Limitations

- The "best" model, in our context, is one that balances precision, recall, and F1 scores while mitigating inherent challenges. Through rigorous evaluation, the ultimate judgment points to the Enhanced Stance Classifier, a fusion of **bidirectional Long Short-Term Memory** (LSTM) layers and **'talkheads_ggelu_bert_en_base' pre-trained BERT embeddings**, as the prime candidate for real-world application.
- Evaluative metrics serve as the compass guiding our judgment. This model attains a remarkable precision of 56%, an invaluable attribute for accurate classification. Its **recall** rate of **57.05%** ensures that the majority of relevant instances are captured. The **F1 scores**, a harmony of **precision(57.50%)** and recall, are at **52% (macro)** and **57.021% (weighted)**, reiterating the model's efficiency. I have used **independent test data** for my evaluation due to the data quality issues. I have got an average F1 score (macro) of 40%.
- While our Enhanced Stance Classifier exhibits remarkable potential, it's essential to acknowledge its **limitations**. The model, despite its prowess, confronts the intricacies of textual nuances in political discourse, **primarily owing to data quality constraints**. Although it provides commendable results, there's room for improvement. The subtleties of political opinions often evade simple classification models. Further research and feature engineering may help enhance its adaptability to nuanced linguistic nuances, making it a more reliable tool for real-world political stance classification on twitter social media.

References:

- *A dataset for multi-target stance detection | request PDF - researchgate*. Available at: https://www.researchgate.net/publication/318741766_A_Dataset_for_Multi-Target_Stance_Detection

Appendices:

Fig1: Classifier Model: No improvement in Training and validation Accuracy after reaching around 54%

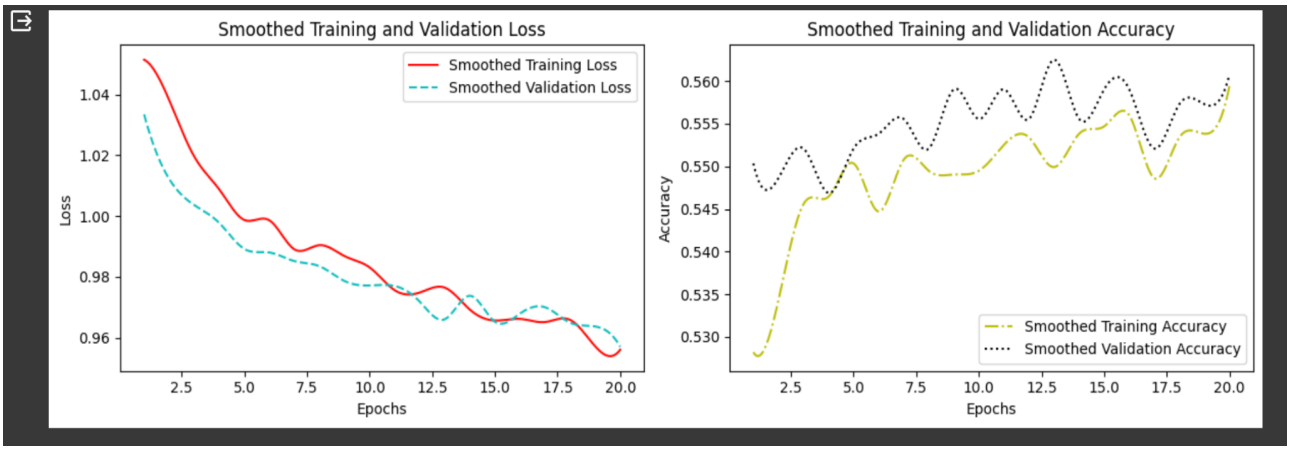

Fig: 2A: Bidirectional LSTM Model with Bert Embeddings

No Hyperparameter Tuning:

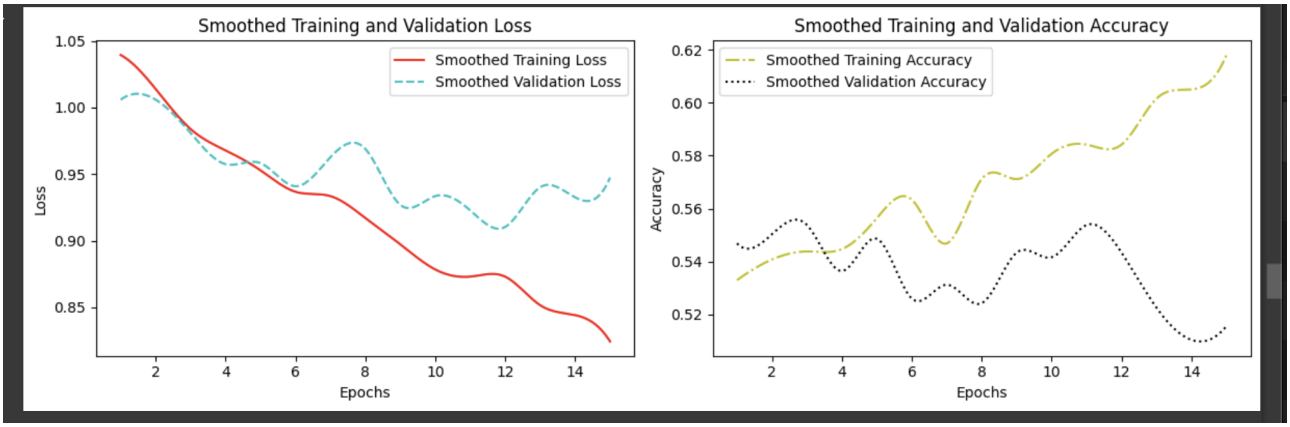

Fig: 2B: Bidirectional LSTM Model with Bert Embeddings

HyperparameterTuning: Data Augmentation, Regularization, Optimizer and Learning Rate.

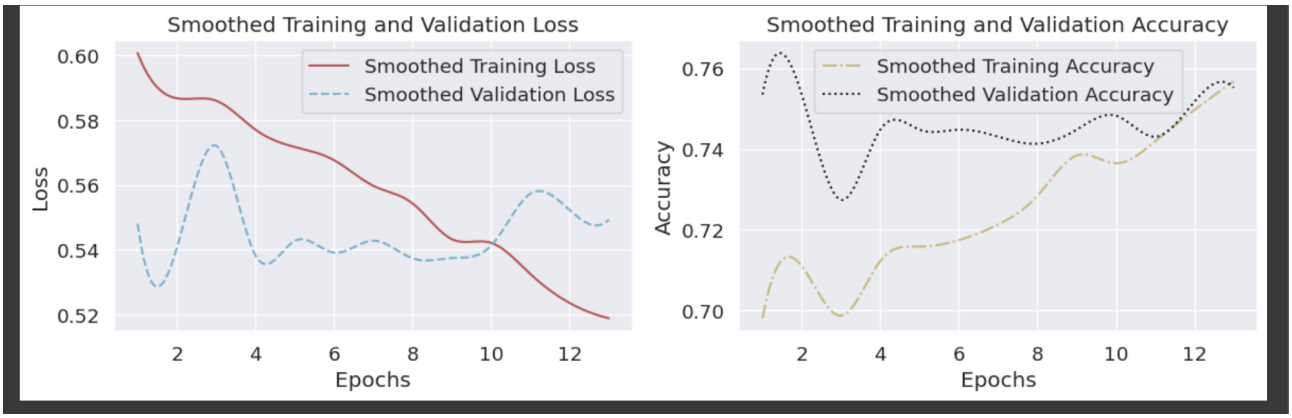

Table: Training and Validation:

Evaluation Frame	**BaseLine Classifier Model** with Small Bert Embedding	**Bidirectional LSTM**-Enhanced Stance Classifier with BERT Embeddings
Training Categorical Accuracy	0.55	0.72
Val Categorical Training Accuracy	0.54	0.59
Training Precision Accuracy	0.60	0.74
Val precision Accuracy	0.54	0.60

Classification Report:

```
Evaluation Results:
Precision : 0.5756521614691554
Recall : 0.5705128205128205
F1 (Weighted): 0.5721281094574027
F1 (Macro): 0.4938789445447007

Classification Report:
              precision    recall  f1-score   support

           0       0.34      0.41      0.37       230
           1       0.43      0.38      0.40       303
           2       0.71      0.70      0.71       715

    accuracy                           0.57      1248
   macro avg       0.49      0.50      0.49      1248
weighted avg       0.58      0.57      0.57      1248
```

Confusion Matrix:
Stance Classes: [NONE: 0, FAVOR: 1, AGAINST:2]

```
[[ 94  45  91]
 [ 78 114 111]
 [103 108 504]]
```