# Deep Learning (COSC 2779) Assignment 2 Sequence Processing with Deep learning [Tweet Stance Classification]

JYOTI
S3880522
RMIT University
October 16, 2023

## Introduction and Background

Stance classification is a specialized subfield of opinion mining that focuses on automatically determining whether a piece of text expresses support or opposition toward a specific target or proposition. It finds extensive applications in information retrieval, text summarization, and textual entailment. Distinction must be made between stance classification and sentiment analysis. While sentiment analysis assesses the overall positivity, negativity, or neutrality of text, stance classification contextualizes the text with respect to a target topic. Understanding stance is vital for various real-world applications, including tailoring information retrieval results, summarizing text with diverse opinions, assessing logical relationships between texts, and analyzing public sentiment.

## Objective and Problem Definition

The objective of project is to create a deep learning model for classification of tweets centered around five politically-charged topics: "Atheism," "the Feminist Movement," "Climate Change is a Real Concern," "Legalization of Abortion," and "Hillary Clinton." The model's task is to categorize each tweet into one of three classes: "FAVOUR," indicating support for the target; "AGAINST," indicating opposition to the target; or "NEITHER," signifying a neutral stance.

The original dataset used in this project for stance classification of tweets is sourced from the "Semeval-2016 Task 6: Detecting Stance in Tweets "developed by Saif M. Mohammad and his team and was introduced as part of the International Workshop on Semantic Evaluation (SemEval-16) in June 2016 in a series of workshops that focus on various natural language processing and computational linguistics tasks.

## EDA: Exploratory Data Analysis

**Training Dataset:** Null values: There are no null values in the training dataset for any of the columns (Tweet, Target, Stance, Opinion Towards, Sentiment).There are 2 duplicated rows in the training dataset."AGAINST" stance has 1395 instances. "NONE" stance has 766 instances. "FAVOR" stance has 753 instances.
**Test Dataset:**Null values: There are no null values in the test dataset for any of the columns (Tweet, Target, Stance, Opinion Towards, Sentiment). Duplicated rows: There are no duplicated rows in the test dataset."AGAINST" stance has 715 instances. "FAVOR" stance has 304 instances. "NONE" stance has 230 instances.

## EDA Approach & Data Processing Approach

- Data preprocessing and cleaning were initiated, beginning with a shuffle of the dataset to enhance model training speed and reduce potential biases.
- Extensive exploratory data analysis was conducted on both training and testing datasets, providing insights into the tweet data followed by an examination of dataset statistics revealed no missing values, ensuring data completeness and reliability.
- Class imbalance was identified when grouping data by the "Target" variable, with "Climate Change is a Real Concern" having the fewest tweets (395), while other targets ranged from 510 to 690 tweets.
- The distribution of tweets within different targets and stances varied significantly, with some categories having more "AGAINST" tweets than others.

**Data Preprocessing:** I conducted a comprehensive text preprocessing on Twitter data and generated word clouds for different target classes. Key steps include lowercasing, special character and hashtag removal, symbol and digit replacement, and non-ASCII character removal. Custom stop words, such as "semst," are excluded to enhance data quality. The code culminates in visually representing the most frequent keywords in the tweets to ensures data quality, readability, and meaningful insights, making it suitable for text analysis and modelling.A detailed appendix is provided, summarizing the distribution of tweets across target categories and stances.

## Model 1 – BERT (Bidirectional Encoder Representations from Transformers)

According to Devlin et al. (2019), BERT is a bidirectional model based on the Transformer architecture (VASWANI et al., 2017) that replaces the sequential nature of recurrent networks with a much faster attention-based approach. BERT works as a Masked-Language model (i.e., a language representation model) allowing to perform different NLP tasks and obtaining state-of-the-art results (KAWINTIRANON; SINGH, 2021; GIORGIONI et al., 2020). Bidirectional Encoder Representations from Transformers, BERT is one of the recent pre-trained models designed by Google (Devlin et al., 2018), which is a bidirectional transformer. However, I have used pre-trained BERT (Large-Uncased) model for stance detection.

**Baseline Model:** I build a BERT-based custom model( small_bert/bert_en_uncased_L-4_H-512_A-8/1" )for tweet stance classification with two BERT encoders for text and target, and a Dense output layer for classification and split the training data into training and validation sets. The model takes two inputs - "tweet" and "target." In tweet stance classification, it's crucial to consider both the content of the tweet and the specific target or topic being discussed. By providing dual inputs, the model can effectively capture the relationship between the tweet and its stance toward the target. Model utilizes the BERT preprocessing layer from TensorFlow Hub to transform the text inputs into numeric token IDs and arrange them into tensors. This is essential for BERT models, as they require standardized text preprocessing. To Encode 'Stance' labels, I used one-hot encoding for classification. After this, I compiled the model with specified parameters, e.g., optimizer, loss function, evaluation metrics and trained the model for a specified number of epochs (5, 10, 20,30) with a defined batch size 32. Afterwards, I make predictions on the validation set to evaluate model, Validation accuracy remains very low around 50%. Later, I computed a classification report with precision, recall, F1-score, and create a confusion matrix to visualize model performance, refer to appendix.

**Run 2 (Hyperparameter Tuning):** Here I experimentd with hyperparameters, such as learning rate and optimizer, to fine-tune the model. Intitally I set the learning rate to 1e-4 and cho ose "Adam" as the optimizer with batch size to 64, epoch - 20. Here, I applied L2 regularization with a strength of 1e-5 to the model's dense layer and implemented early stopping with a patience of 3.
**Run 3 (Data Augmentation):** To handle class imbalance, I augmented the training data using the nlpaug library and BERT-based contextual word embeddings and did model training followed by model evaluation.
**Run 4 (Class Weight Balancing):** To mitigate the impact of class imbalance, I assigned different weights to each class during training. This is done to give more importance to the minority classes. Keras allows to specify class weights when compiling your model, which do influence the loss I added more importance to the minority classes during training to improve the model's performance on imbalanced datasets. Likewise, model is trained and evaluated, however, not major improvement in accuracy.
**Run 5 (Oversampling of Data with SMOTE ):** The imbalanced-learn library's SMOTE is used to perform oversampling. SMOTE generates synthetic samples for the minority class (in this case, underrepresented stances) to balance the class distribution. The sampling_strategy parameter is set to 'auto' to balance the classes automatically.The oversampling step is included in a machine learning pipeline to ensure that it's applied before training the model. The model pipeline is trained with the training data and validation data. The oversampled training data is used during training. It was taking a lot of computation units too, and not able to secure proper results.

## Model 2 ( LSTM pre-trained GloVe word embeddings and Bidirectional LSTM layers)

As the performance was not very up to mark with BERT, The deep learning model I experimented with second model which uses LSTM layers for target classification . The model architecture is as follows:
**Embedding Layer:** The model starts with an embedding layer that takes the tokenized input sequences and maps them to dense vector representations. It uses pre-trained GloVe word embeddings to initialize these vectors. The weights in this layer are set as non-trainable to keep the embeddings fixed during initial training.
**Bidirectional LSTM Layers:** The embedding layer is followed by bidirectional Long Short-Term Memory (LSTM) layers. LSTM is a type of recurrent neural network (RNN) that can capture sequential patterns in the data. Bidirectional LSTMs read input sequences in both forward and backward directions, allowing to capture dependencies in both directions.
**Flatten Layer:** After LSTM layers, flatten layer is added to convert multidimensional output into one-dimensional vector.
**Dense Layers:** Two dense layers are added for further feature extraction and classification. The first dense layer has ReLU activation and dropout regularization to prevent overfitting. The second dense layer has five units (equal to the number of target classes) and uses softmax activation for target classification.

## Evaluation Approach of Model 2

**Model Compilation:** The model is compiled with the categorical cross-entropy loss function for multi-class classification. The Adam optimizer is used for gradient-based optimization. Custom metric called f1_m to calculate F1 score.
**Hyperparameter Tuning**: I performed hyperparameter tuning for the target model. Hyperparameters tested include the regularization strength (reg_lambda) and the dropout rate (drop) with different combinations.
**Model Training and Fine tuning:** The best model configuration, based on the hyperparameter tuning results, is selected for target classification.The first layer of the target model (the embedding layer) is unfrozen for fine-tuning. The learning rate is adjusted, and the model is trained for an additional five epochs to adapt to the specific task of stance detection.
**Unseen Data Prediction:**After training the model, the code makes predictions on unseen data, which is the test set, for stance classification.
**Output Metrics:** The code evaluates the model's performance using macro F1 score, which is a measure of classification accuracy that considers precision and recall for each class. It also provides a classification report, which includes precision, recall, and F1 score for each class (AGAINST, FAVOR, NONE) in the stance predictions.

**Results and Experiments Analysis for Model 1 ( BERT) :** The model is evaluated on the test data with an accuracy of 57%. However, the classification report shows low precision, recall, and F1-Score for classes other than "AGAINST." The baseline BERT model performed poorly, with low accuracy and F1-Score.Hyperparameter tuning improved accuracy but didn't address the low F1-Score for some classes. Data augmentation and class weight balancing had similar results to the tuned models.The models tend to predict "AGAINST" well but struggle with other classes.

| Run | Hyperparameters | Accuracy | Precision AGAINST | Recall AGAINST | F1-Score AGAINST | Precision FAVOR | Recall FAVOR | F1-Score FAVOR | Precision (NEITHER) | Recall (NEITHER) | F1-Score (NEITHER) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Baseline | Epochs: 10 | 26% | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.26 | 1.00 | 0.41 |
| Run 1 | Epochs: 15, Learning Rate: 1e-4 | 48% | 0.48 | 1.00 | 0.65 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| Run 2 | Epochs: 20, Learning Rate: 1e-4 | 48% | 0.48 | 1.00 | 0.65 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| Run 4 | Epochs: 30, Learning Rate: 1e-4 | 48% | 0.48 | 1.00 | 0.65 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| Test | Test Data Evaluation | 57% | 0.57 | 1.00 | 0.73 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

Tab le 1 : Model 1 Results Analysis

Model 2 [ LSTM ]: Model Performance [**Accuracy**: 60.69%]

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Class 0 | 89.65% | 62.72% | 73.81% | 1022 |
| Class 1 | 27.83% | 39.02% | 32.49% | 164 |
| Class 2 | 17.43% | 84.13% | 28.88% | 63 |

Table 2 : LSTM Model Results Analysis

| Class | Macro Avg | Weighted Avg |
|---|---|---|
| Precision | 44.97% | 77.89% |
| Recall | 61.96% | 60.69% |
| F1-Score | 45.06% | 66.11% |
| Support | 1249 | 1249 |

Table 3: Overall Metrics

**Ultimate judgment and conclusion Model 1 :** The baseline BERT model performed poorly, with low accuracy and F1-Score. Hyperparameter tuning improved accuracy but didn't address the low F1-Score for some classes.Data augmentation and class weight balancing had similar results to the tuned models.The models tend to predict "AGAINST" well but struggle with other classes.There is room for further improvement, possibly by refining the model architecture, incorporating more data, or exploring advanced techniques.

**Ultimate judgment and conclusion Summary Model 2:** The target classification model achieved an F1-Score of around 1.6726 (training) and 1.6032 (validation) for the best hyperparametersThe stance classification model achieved an F1-Score of approximately 1.1132 (training) and 1.0711 (validation) for the best hyperparameters.The macro F1-Score for the stance model on unseen data was approximately 0.6069The models were evaluated on unseen data.The F1-Score, particularly the macro F1-Score, was reported as a performance metric.The macro F1-Score for the stance model was reported to be approximately 0.6069.

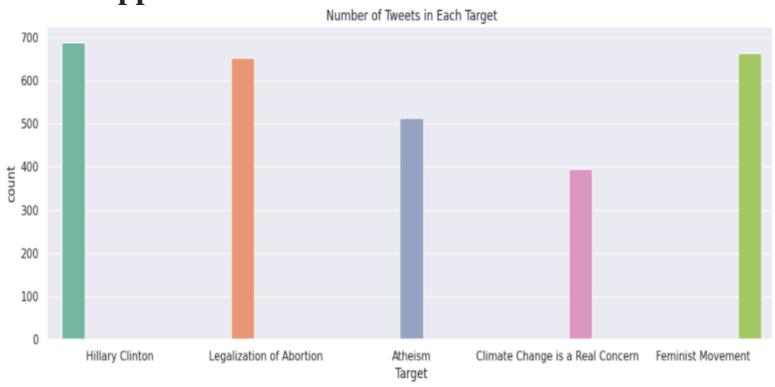
**Conclusion and Future work:** In summary, my findings indicate that the LSTM model demonstrated superior performance in the context of tweet stance classification compared to the BERT model. The LSTM model was optimized with specific hyperparameters: it benefitted from having 128 units for both the LSTM and Bidirectional LSTM layers, a dropout and recurrent dropout ratio of 0.2, and L2 regularization at 0.0001. Additionally, the model's performance was maximized with an Adam optimizer learning rate of 0.001. Moreover, it is worth noting that the tweet data available for this task contained a relatively limited amount of textual content, which could have impacted the model's ability to achieve optimal performance. Thus, drawing robust conclusions from our model's performance may be somewhat constrained by data limitations.

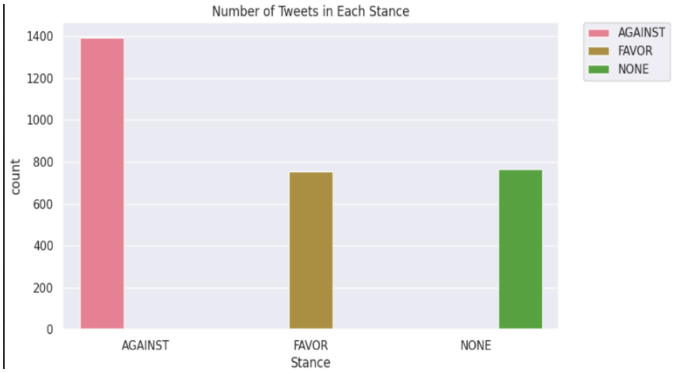
In conclusion, the LSTM model emerged as the more effective choice for tweet stance classification. Still, the success of our approach could significantly benefit from improved data density, offering the potential to design more advanced architectures and achieve higher accuracies across various stance classes.

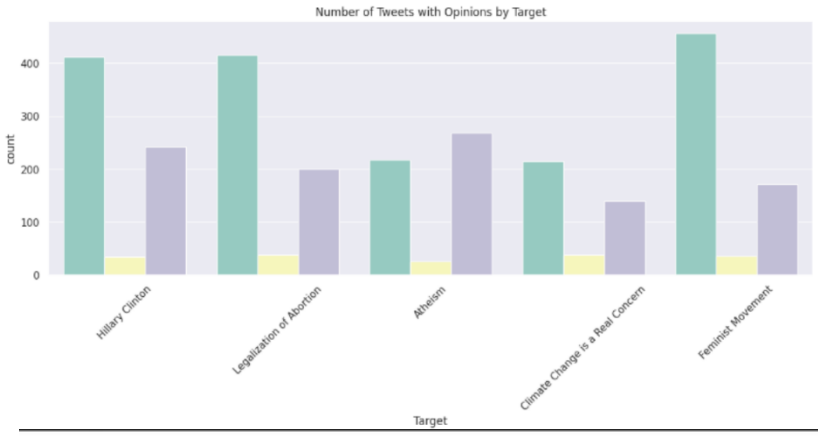
## References

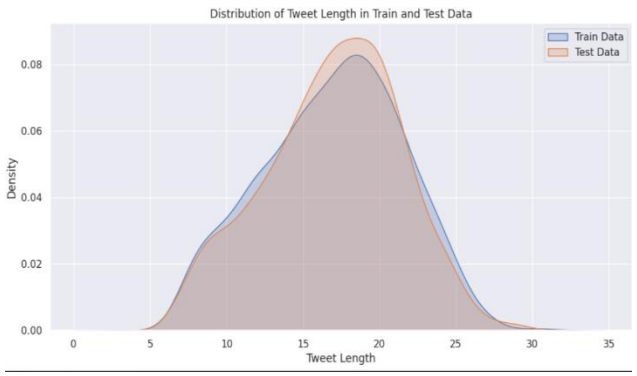
[1] Zarrella, G. and Marsh, A., 2016. Mitre at semeval-2016 task 6: Transfer learning for stance detection. arXiv preprint arXiv:1606.03784.
[2] Du, J., Xu, R., He, Y. and Gui, L., 2017, August. Stance classification with tar get-specific neural attention networks. International Joint Conferences on Artificial Intelligence
[3]Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting Stance in Tweets. In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pages 31–41, San Diego, California. Association for Computational Linguistics
[4] RMIT Canvas labs for Deep Learning

## Appendix

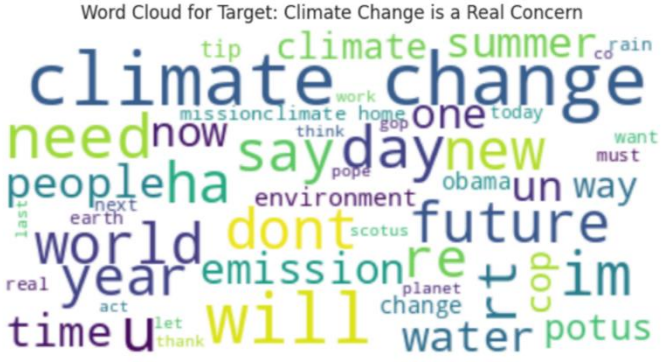


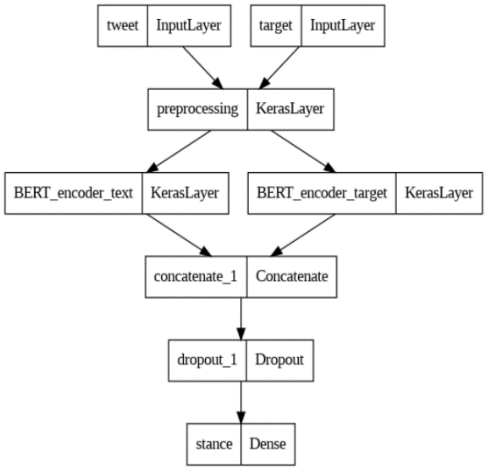




1. The tweet explicitly expresses opinion about the target, a part of the target, or an aspect of the target.
3. The tweet is not explicitly expressing opinion. (For example, the tweet is simply giving information.)
2. The tweet does NOT expresses opinion about the target but it HAS opinion about something or someone other than the target.

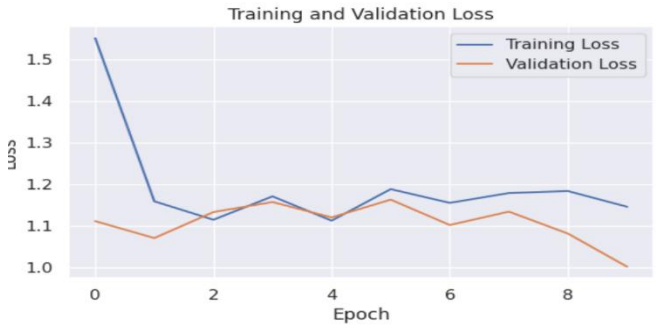


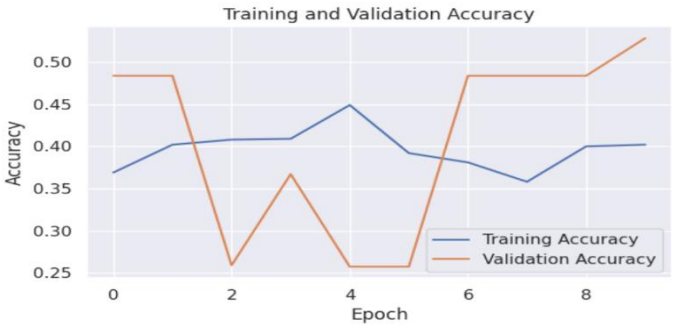

Model 1 : BERT Architecture (Baseline and fine tuning results)



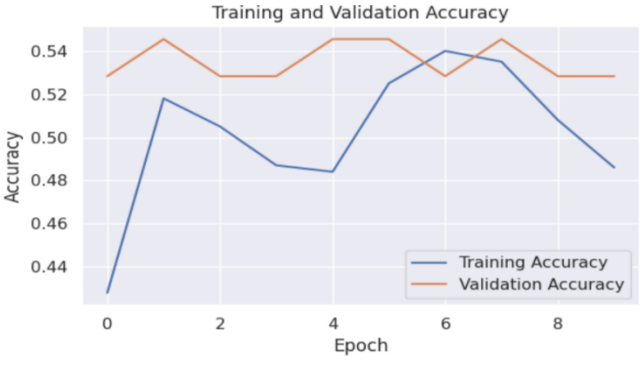

```
19/19 [==============================] - 5s 230ms/step
Classification Report:
              precision    recall  f1-score   support

     AGAINST       0.54      0.99      0.70       282
       FAVOR       0.41      0.19      0.26       151
     NEITHER       1.00      0.00      0.00       150

    accuracy                           0.53       583
   macro avg       0.65      0.39      0.32       583
weighted avg       0.63      0.53      0.41       583

Confusion Matrix:
 [[279   3   0]
 [122  29   0]
 [111  39   0]]
```

```
19/19 [==============================] - 5s 249ms/step
Classification Report:
              precision    recall  f1-score   support

     AGAINST       0.54      0.99      0.70       282
       FAVOR       0.41      0.19      0.26       151
     NEITHER       1.00      0.00      0.00       150

    accuracy                           0.53       583
   macro avg       0.65      0.39      0.32       583
weighted avg       0.63      0.53      0.41       583

Confusion Matrix:
 [[279   3   0]
 [122  29   0]
 [111  39   0]]
```








Seocnd Model : 1 ) Training Results w diff params

Latest Scores:

| | l_train | l_val | reg | dropo | val_acc |
|---|---|---|---|---|---|
| 0 | 1.730123 | 1.609754 | 0.0050 | 0.7 | 0.118883 |
| 1 | 1.851414 | 1.707038 | 0.0050 | 0.8 | 0.082217 |
| 2 | 1.616411 | 1.504248 | 0.0010 | 0.7 | 0.310241 |
| 3 | 1.745965 | 1.620335 | 0.0010 | 0.8 | 0.105868 |
| 4 | 1.517826 | 1.417371 | 0.0005 | 0.7 | 0.282285 |
| 5 | 1.682845 | 1.582138 | 0.0005 | 0.8 | 0.047096 |

Best Scores:

| | l_train | l_val | reg | dropo | val_acc |
|---|---|---|---|---|---|
| 0 | 1.730123 | 1.609754 | 0.0050 | 0.7 | 0.118883 |
| 1 | 1.851414 | 1.707038 | 0.0050 | 0.8 | 0.082217 |
| 2 | 1.616411 | 1.504248 | 0.0010 | 0.7 | 0.310241 |
| 3 | 1.745965 | 1.620335 | 0.0010 | 0.8 | 0.105868 |
| 4 | 1.517826 | 1.417371 | 0.0005 | 0.7 | 0.282285 |
| 5 | 1.682845 | 1.582138 | 0.0005 | 0.8 | 0.047096 |

2) Test Results on unseen data

```
#Print macro F1 score
print(f1_m(pred,y_teststance))
```

```
{tf.Tensor(0.60688543, shape=(), dtype=float32)
 'f1-score': 0.7380541162924582,
 'support': 1022},
'1': {'precision': 0.2782608695652174,
 'recall': 0.3902439024390244,
 'f1-score': 0.3248730964467005,
 'support': 164},
'2': {'precision': 0.17434210526315788,
 'recall': 0.8412698412698413,
 'f1-score': 0.2888283378746594,
 'support': 63},
'accuracy': 0.6068855084067254,
'macro avg': {'precision': 0.44970215711062395,
 'recall': 0.6195717697555319,
 'f1-score': 0.45058518353793936,
 'support': 1249},
'weighted avg': {'precision': 0.7788990461704147,
 'recall': 0.6068855084067254,
 'f1-score': 0.6611422577696194,
 'support': 1249}}
```