# CSE440: Natural Language Processing II
# Project Report

**Section: 01**

**Group: 11**

## Group Members:

| **Name** | **ID** |
|---|---|
| Nafees Raiyan Rahman | 21301315 |
| Nafisa Mehreen | 23241114 |
| Afridi Alam Polock | 21301303 |
| Lamia Alam Emu | 21301662 |

## Introduction

The development of NLP (natural language processing) has sparked advancements in various domains. For example: machine translation, sentiment analysis and text categorization. Out of all these techniques, LSTM (Long Short-Term Memory) became known for modeling sequential data due to their ability to capture long-range dependencies. We had to do two tasks in this project which were sentiment analysis in task A and text generation in task B. For task A, we used LSTM networks which enhanced the model's ability to capture contextual dependencies. Task B challenged us to generate contextually relevant responses to given prompts, leveraging Bidirectional LSTM for improved understanding of language semantics. In the subsequent sections, we showcase these model's performance, employing metrics  such as accuracy. By accomplishing these steps, our goal is to build an effective sentiment classification model and gain insights into the impact of incorporating LSTM layers on model performance.

## Data loading and preprocessing

The dataset is in .csv format. It consists of collections of texts, each labeled with sentiments indicating whether the text is sexist or non sexist and also categorize the text. The dataset is initially splitted into **70-30** ratios for training and testing. We splitted 30 percent of the training dataset for validation. As a result, the training data teaches the model to recognize patterns and relationships within the data and the validation data plays a vital role in fine-tuning the model when it becomes too specialized in the training data.

Next, we applied some libraries such as Keras Tokenizer to tokenize and convert text reviews into sequences of integers. It establishes a mapping between words and their corresponding indices. The tokenizer is fitted on the training set and is applied to both training and test sets. We have applied padding to both the training and test sets to ensure uniform sequence lengths. We also used nltk and matplotlib. We also did word embedding with a tensor library. We did a data split for task B.

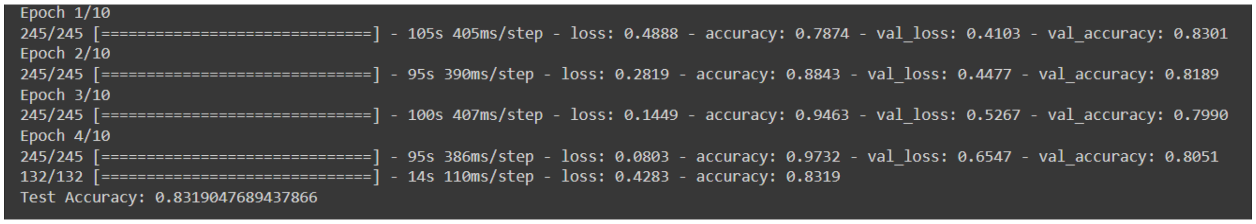
## Training models

**Bidirectional LSTM model:** a variant of the gated recurrence relation is the implementation of Bidirectional LSTM. The model architecture consists of an embedding layer. Following that, a Bi-LSTM layer with **128 units**, followed by a dense layer with a sigmoid activation for binary classification. The Bidirectional LSTM layer captures contextual information from both forward and backward sequences, enhancing its ability to recognize detailed patterns in text data. Consequently, it aims to improve sentiment analysis performance.

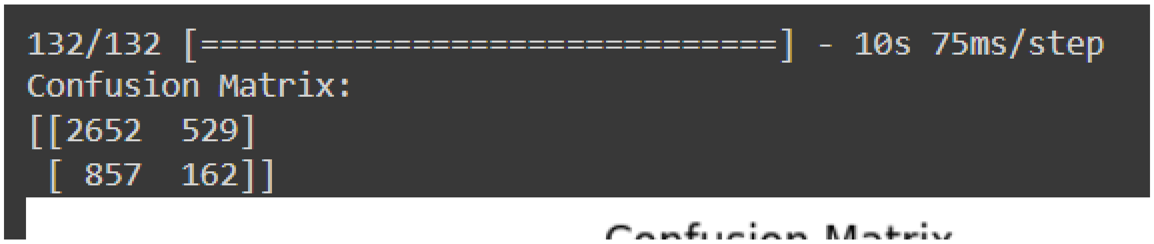
For the model, we have applied **Adam optimizer** to minimize the model's binary cross-entropy loss during training. We have configured the epoch size as 10. We have set the batch size to 32 for the training set.

## Performance analysis

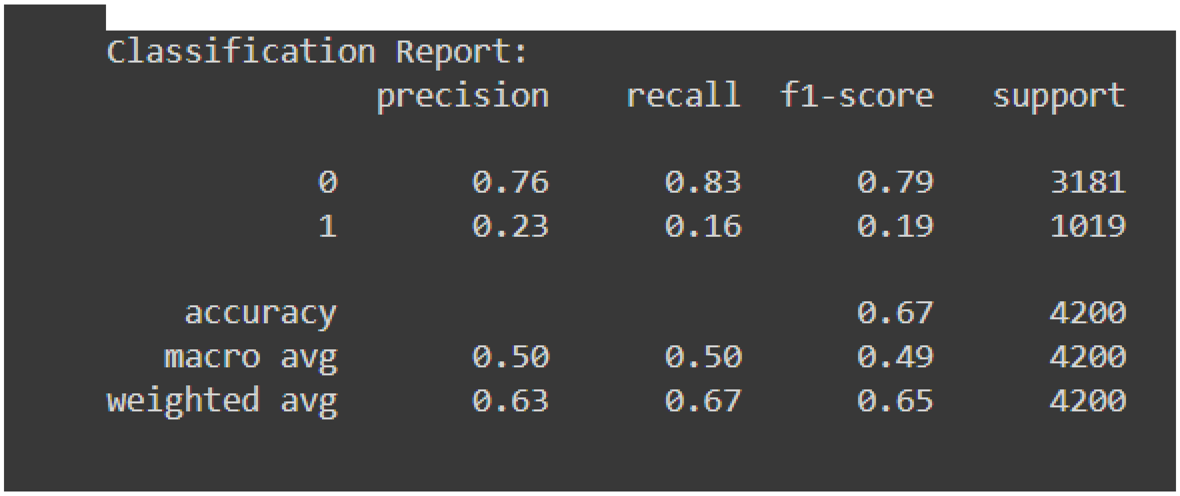
The performance analysis for Task A is-



```
Epoch 1/10
245/245 [==============================] - 105s 405ms/step - loss: 0.4888 - accuracy: 0.7874 - val_loss: 0.4103 - val_accuracy: 0.8301
Epoch 2/10
245/245 [==============================] - 95s 390ms/step - loss: 0.2819 - accuracy: 0.8843 - val_loss: 0.4477 - val_accuracy: 0.8189
Epoch 3/10
245/245 [==============================] - 100s 407ms/step - loss: 0.1449 - accuracy: 0.9463 - val_loss: 0.5267 - val_accuracy: 0.7990
Epoch 4/10
245/245 [==============================] - 95s 386ms/step - loss: 0.0803 - accuracy: 0.9732 - val_loss: 0.6547 - val_accuracy: 0.8051
132/132 [==============================] - 14s 110ms/step - loss: 0.4283 - accuracy: 0.8319
Test Accuracy: 0.8319047689437866
```

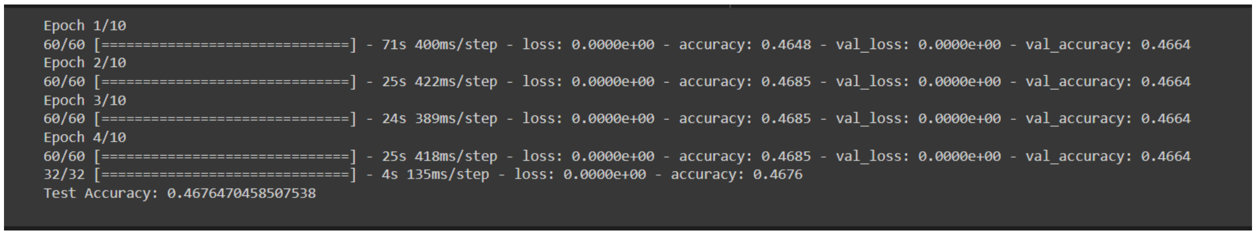


```
132/132 [==============================] - 10s 75ms/step
Confusion Matrix:
[[2652  529]
 [ 857  162]]
```

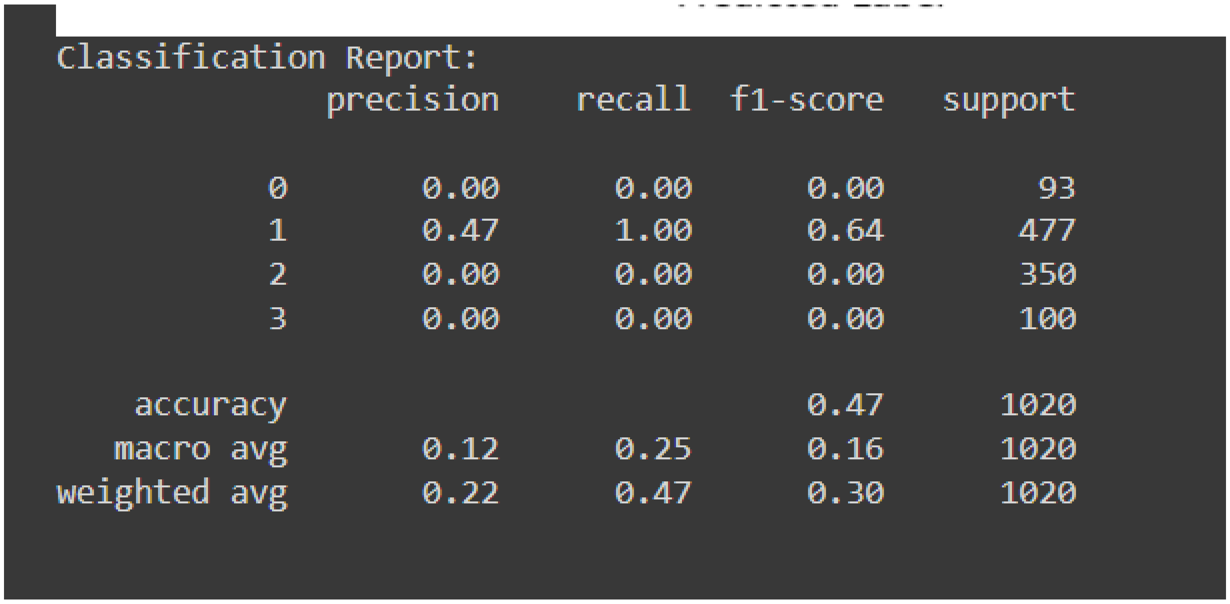


```
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.83      0.79      3181
           1       0.23      0.16      0.19      1019

    accuracy                           0.67      4200
   macro avg       0.50      0.50      0.49      4200
weighted avg       0.63      0.67      0.65      4200
```

The performance analysis for Task B is-



```
Epoch 1/10
60/60 [==============================] - 71s 400ms/step - loss: 0.0000e+00 - accuracy: 0.4648 - val_loss: 0.0000e+00 - val_accuracy: 0.4664
Epoch 2/10
60/60 [==============================] - 25s 422ms/step - loss: 0.0000e+00 - accuracy: 0.4685 - val_loss: 0.0000e+00 - val_accuracy: 0.4664
Epoch 3/10
60/60 [==============================] - 24s 389ms/step - loss: 0.0000e+00 - accuracy: 0.4685 - val_loss: 0.0000e+00 - val_accuracy: 0.4664
Epoch 4/10
60/60 [==============================] - 25s 418ms/step - loss: 0.0000e+00 - accuracy: 0.4685 - val_loss: 0.0000e+00 - val_accuracy: 0.4664
32/32 [==============================] - 4s 135ms/step - loss: 0.0000e+00 - accuracy: 0.4676
Test Accuracy: 0.4676470458507538
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        93
           1       0.47      1.00      0.64       477
           2       0.00      0.00      0.00       350
           3       0.00      0.00      0.00       100

    accuracy                           0.47      1020
   macro avg       0.12      0.25      0.16      1020
weighted avg       0.22      0.47      0.30      1020
```

## Improvements

A couple of improvements to enhance model performance:

1. **Early stopping and increasing epochs:** From the graph, we observe that the model is mildly overfitting in task A and overfitting in task B. In this situation, if we integrate early stopping, we can also stop the training when the improvement is not increasing and save the best weights. Also, increasing the number of epochs may boost the performance of the model.
2. **Integrate GRU instead of BLSTM:** GRU is often faster than BLSTM as it has one less gate and no cell state to update. So, it may show a better performance.
3. **Dropout for overfitting:** Implementing 10%-25% dropout can help address overfitting issues which will help the model to train well.
4. **Increase padding size:** In task B, it seems like the BLSTM model isn't showing better results. So, calculating the max length based on other parameters food padding could improve its performance.

## Conclusion

In conclusion, in task A, the model achieved relatively good accuracy on the test set. The use of BLSTM with an embedding layer allows capturing contextual information effectively. The model's performance might benefit from experimenting with different hyperparameters such as LSTM units and augmenting the dataset could help generalize the model better. And in task B, the choice of loss function seemed suitable for multi-class classification, but it had to match the output encoding. Exploring different architectures like adding more layers or utilizing attention mechanisms could enhance model performance. In our opinion, model performance could be further enhanced by hyperparameter tuning and architectural modifications.