ONLINE SEXISM DETECTION

Sanjana Sabah Khan

ID: 18101502

Md. Abrar Faiaz Khan

ID: 19301106

Mashrur Safir Shabab

ID: 20241037

INTRODUCTION

The Internet and other forms of online communication have become essential to our daily lives, allowing people who would have been completely inaccessible to one another to form relationships. However, with the growth of the internet, there has been a rise in online sexism, which is expected to have a terrible effect on both people and society as a whole. In our project, we will use Long Short-Term Memory (LSTM) neural networks to identify online sexism from a given dataset. We will train our LSTM model on a large corpus of sexist and non-sexist comments, and then use the trained model to detect sexist language in new comments. This text discusses two hierarchical subtasks: Binary Sexism and Category of Sexism.

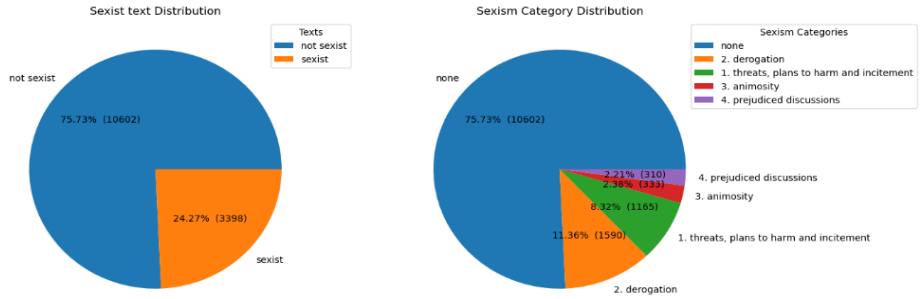
The first task makes a two-class classification where the system has to predict whether the post is sexist or not sexist. The second level disaggregates sexist content into four distinct categories: Threats, Derogation, Animosity, and Prejudiced Discussion. The results will show if the LSTM model can detect sexist language with high accuracy or not.

DATASET DESCRIPTION

The data set was taken from a paper named Explainable Detection of Online Sexism (Kirk et al., 2023). They collected the data from Gab and Reddit and then further processed it.

Task A: For this task, we have used the label_sexist column from the dataset. We have a total of 14000 data points in that column. Among them, 3398 are sexist data points and conversely 10602 are non-sexist data points. This implies an imbalance of the data set where the non-sexist class has 7204 more data sets than its counterpart.

Task B: For task B we used the label_category column from the dataset. For the category of sexism, among 3398 sexist data there are a total of 310 data classed as threats, 333 data sets on prejudiced discussion, 1165 on animosity and 1590 on derogation. In this classification, the animosity and derogation data set are way more compared to threats and prejudiced discussion class. Thus, this creates an imbalance.



While cleaning the data, expressions such as *https://.∗,* ‘’ were removed. The punctuations

were deliberately not touched as the model performs better with them. All the characters were also made small. Unnecessary coloumns like “rewire_id" were dropped as it was not necessary for our task. No null rows and columns were found while cleaning the data. The dataset was splitted in three parts where the trained set consisted of 80% of the data (11200 data points), the test data consisted of 20% and the validation consisted of 10% (1400 data points). In text tokenization all the unique words and assigned with some unique numbers where the size of the vocabulary was 19986. After text tokenization we are padding the tokenized text where we are setting the maximum length of padding 50, and after padding we are getting the dimension for (x_train.shape[0], x_train.shape[1]) is [11200, 50]. For (x_val.shape[0], x_val.shape[1]), the dimension is [1400, 50] and for (x_test.shape[0], x_test.shape[1]) we are getting the dimension of [1400, 50]. The embedding matrix was created with GloVe embedding, more specifically it was created with glove.6b.100d. The dimension of this particular embedding was 100. The amount of nonzero embedding vectors were 0.8003102171520065.

METHODOLOGY

LSTM has 3 gates by which it operates. 1) Forget gate, 2) Input gate, 3) Output gate.

The first gate is the forget gate where LSTM has to decide whether new information will pass or it should carry the previous information. The equation for this gate is:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

Then we have an input gate. Input gate decides which information should be stored in the cell state. It has 2 layers. i_t decides which values will be updated and a tanh layer creates a vector for new information that could be added to the cell state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$
$$C'_t = tanh(W_c.[h_{t-1}, x_t] + b_c)$$

Next we update the old cell state with the new value.

$$C_t = f_t * C_{t-1} + i_t * C'_t$$

Finally, we have the output gate. The sigmoid function decides which part of the cell state will be the output and then it passes through tanh and multiply it by the output of the sigmoid gate.

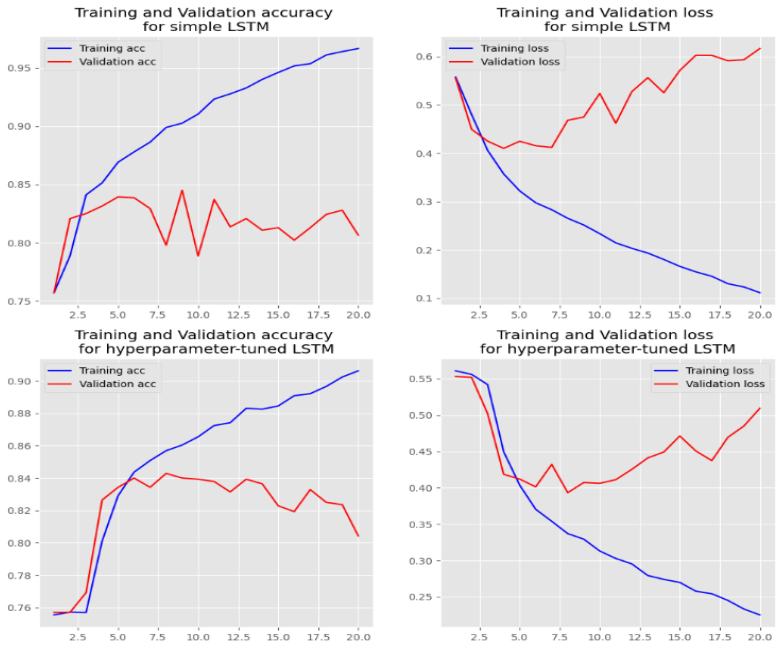
$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$
$$C_t = o_t * tanh(C_t)$$

In our project, we first use the simple LSTM structure. In the code, Lstm_simple() defines a simple LSTM model with one LSTM layer, one Dense layer, and a sigmoid activation function. It uses an embedding layer to transform the input sequence into a fixed-length

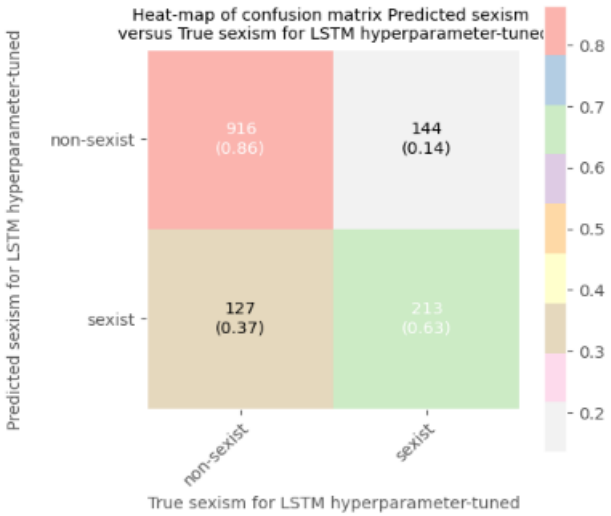
vector representation. The LSTM layer has lstm_out units and a dropout rate of 0.2. The model is compiled using binary cross-entropy, Adam, and accuracy. Then it finds the accuracy and loss for training and test data. Next, we use the LSTM model Hyperparameter tuned. Hypertuning is the process of selecting the best set of hyperparameters for a given model in order to optimize its performance. In the code, The function lstm_hypertuned() defines a LSTM model with hyperparameters such as lstm_out, dropout rate, and Dense layer neurons. It can be used as a starting point for hyperparameter tuning by adjusting the values and observing model performance on a validation set. Then we find the accuracy and loss for training and test data in the same way.

RESULT

Task A: For simple LSTM structure, we have set the epochs value 20 and after 20 epochs, we are getting Training Accuracy: 0.9827, and Training Loss: 0.0808. And for the test result, Testing Accuracy: 0.8014, and Testing Loss: 0.6324. Next, for the hyperparameter tuned LSTM model, the epoch size is 20 and the Training Accuracy is 0.9302, Training Loss is 0.1770 and the Testing Accuracy is 0.8064, Testing Loss is 0.4900. Then we plot the Training & Validation loss & accuracy in Graph and the output is:



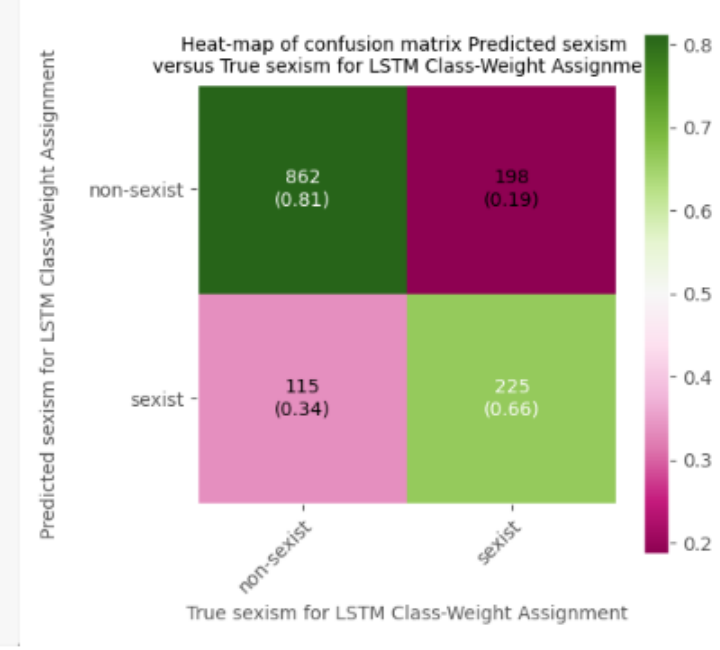
Confusion matrix for hyperparameter tuned LSTM



We are having an overfitting issue here. We use accuracy_classificationrep() function to print the accuracy score and classification report for a given model's prediction on the test data and find the confusion matrix for predicted sexism versus true sexism for LSTM Hyperparameter tuned.

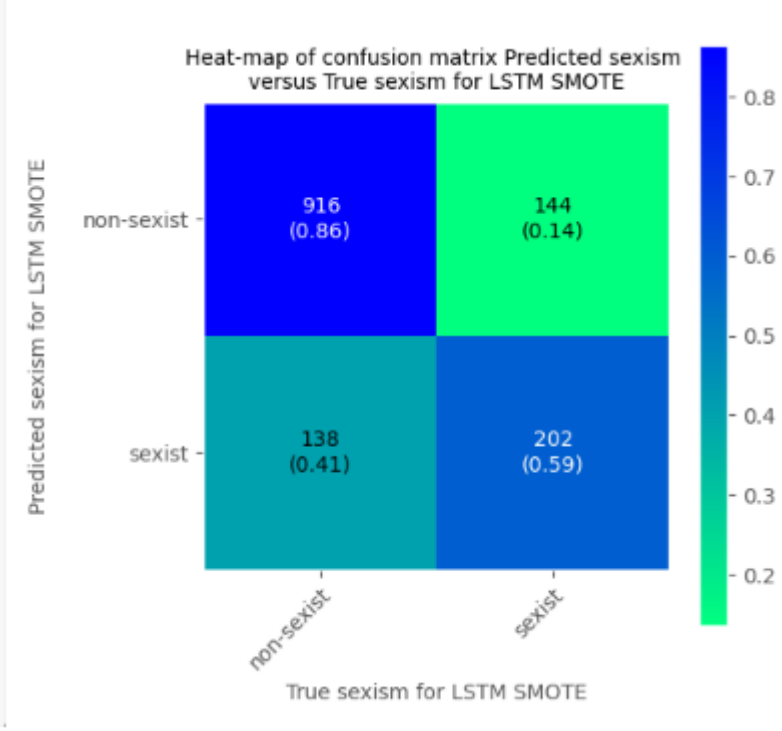
Here we see an imbalance, the sexist class is much lesser in amount than the non sexist class and this imbalance needs to be handled to tackle the overfitting problem. We will handle it in two ways - class weight assignment and oversampling via synthetic data generation using SMOTE.

After Class Weight Assignment, we find training Accuracy: 0.9578, training Loss: 0.1130, testing Accuracy: 0.7764, testing Loss: 0.7397 and LSTM after class-weight assignment accuracy score : 0.7764285714285715.

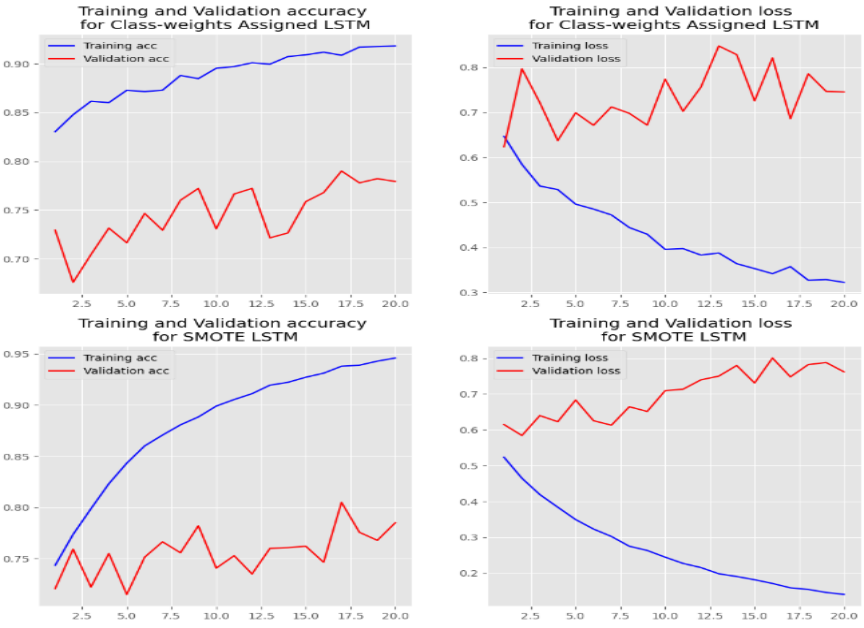


Models	Precision		Recall		F1 Score	
	Sexist	Non-sexist	Sexist	Non-Sexist	Sexist	Non-sexist
Hyperparametertuned LSTM	0.60	0.88	0.63	0.86	0.61	0.87
Class Weight LSTM	0.53	0.88	0.66	0.81	0.59	0.85
SMOTE LSTM	0.58	0.87	0.59	0.86	0.59	0.87

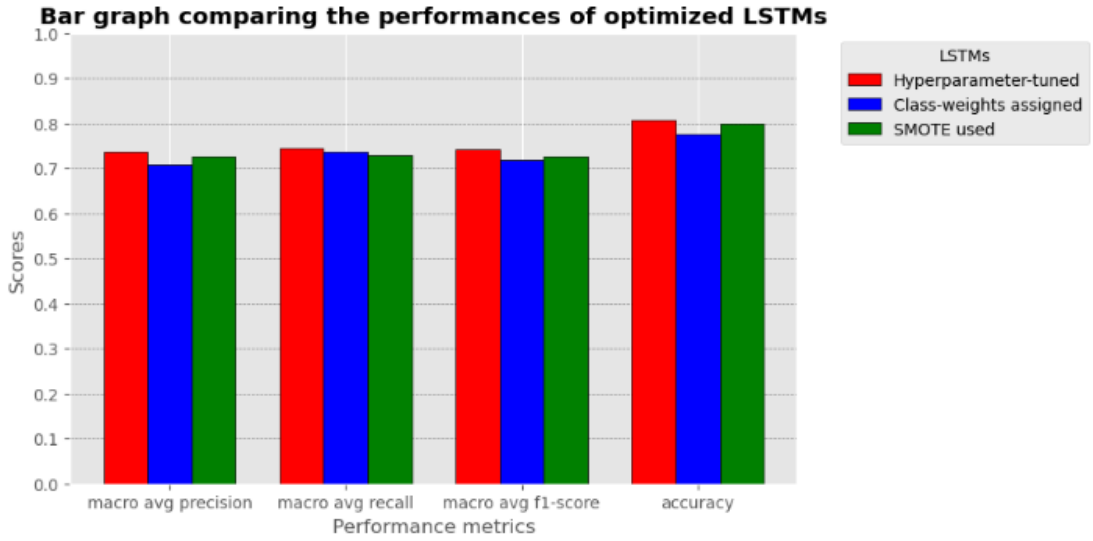
After SMOTE, we are getting the training Accuracy: 0.9848, training Loss: 0.053, testing Accuracy: 0.7986, testing Loss: 0.7505 and LSTM after SMOTE accuracy score : 0.7985714285714286.



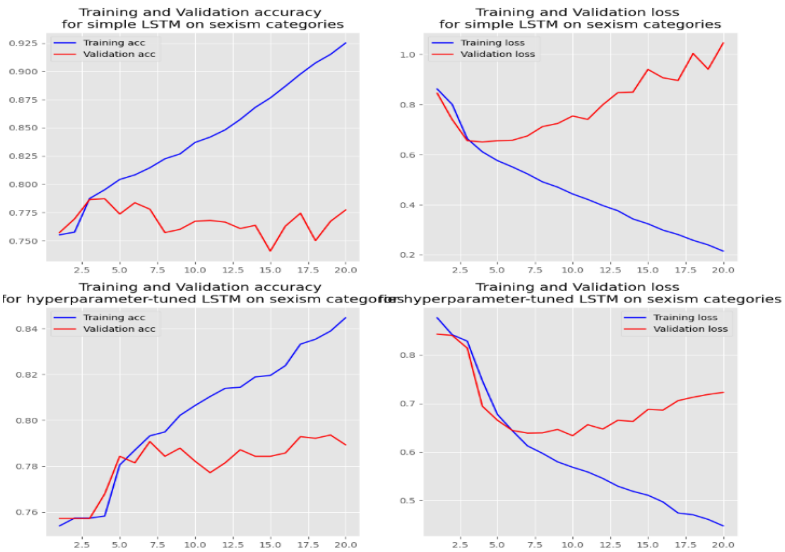
Training & Validation loss & accuracy Graph comparing Class-weight & SMOTE:



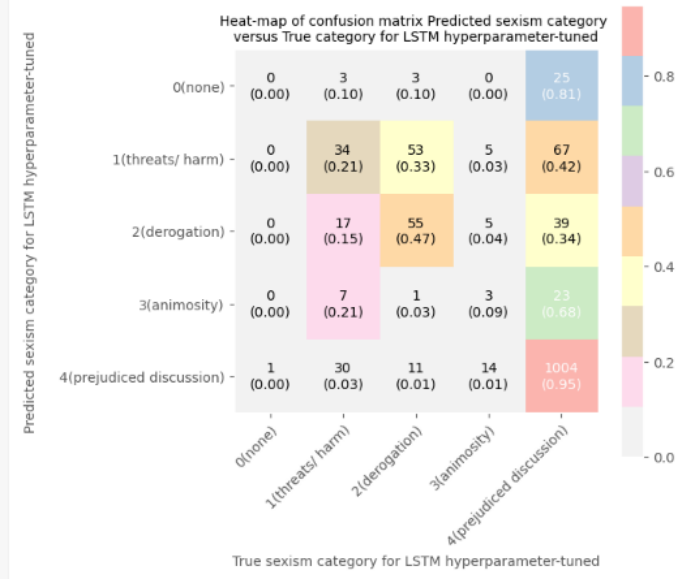
Bargraph showing 4 performance metrics:



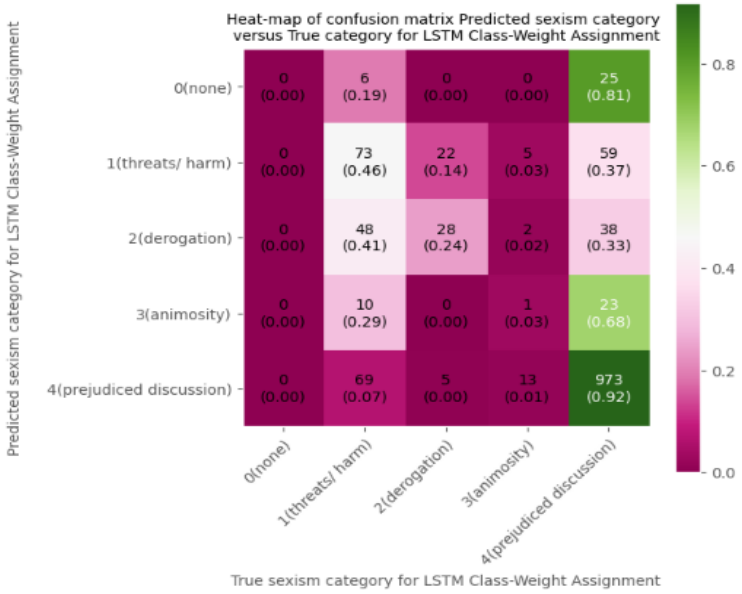
Task B: For simple LSTM structure, we have set the epochs value 20 and after 20 epochs, we are getting Training Accuracy: 0.9436, and Training Loss: 0.1688. And for the test result, Testing Accuracy: 0.7700, and Testing Loss: 1.0397. Next, for the hyperparameter tuned LSTM model, the epoch size is 20 and the Training Accuracy is 0.8586, Training Loss is 0.3720 and the Testing Accuracy is 0.7829, Testing Loss is 0.7149. Then we plot the Training & Validation loss & accuracy in Graph and the output is:



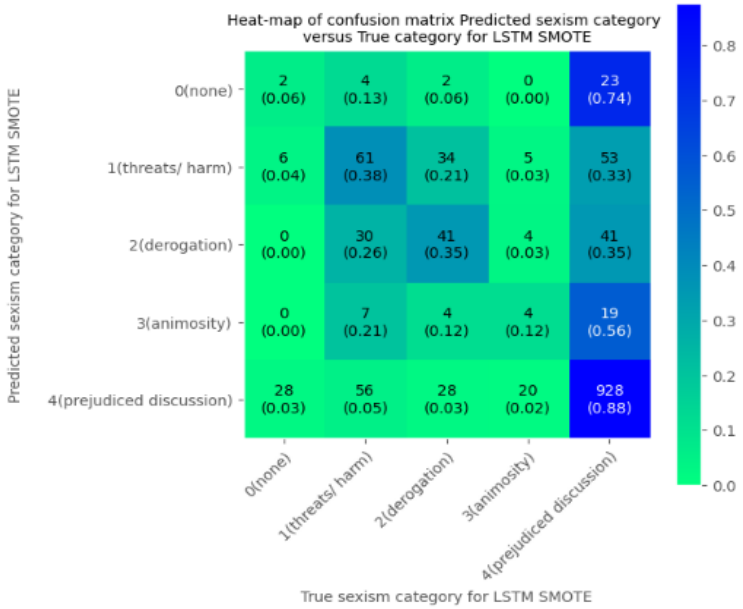
Confusion matrix for hyperparameter tuned LSTM:



We are having an overfitting issue here as well. We see an imbalance, the sexist class is much lesser in amount than the non sexist class and this imbalance needs to be handled to tackle the overfitting problem. We will handle it in two ways - class weight assignment and oversampling via synthetic data generation using SMOTE the way we did in task A. After Class Weight Assignment, we find training Accuracy: 0.9165, training Loss: 0.2724, testing Accuracy: 0.7679, testing Loss: 1.0347 and LSTM after class-weight assignment accuracy score : 0.7678571428571429.

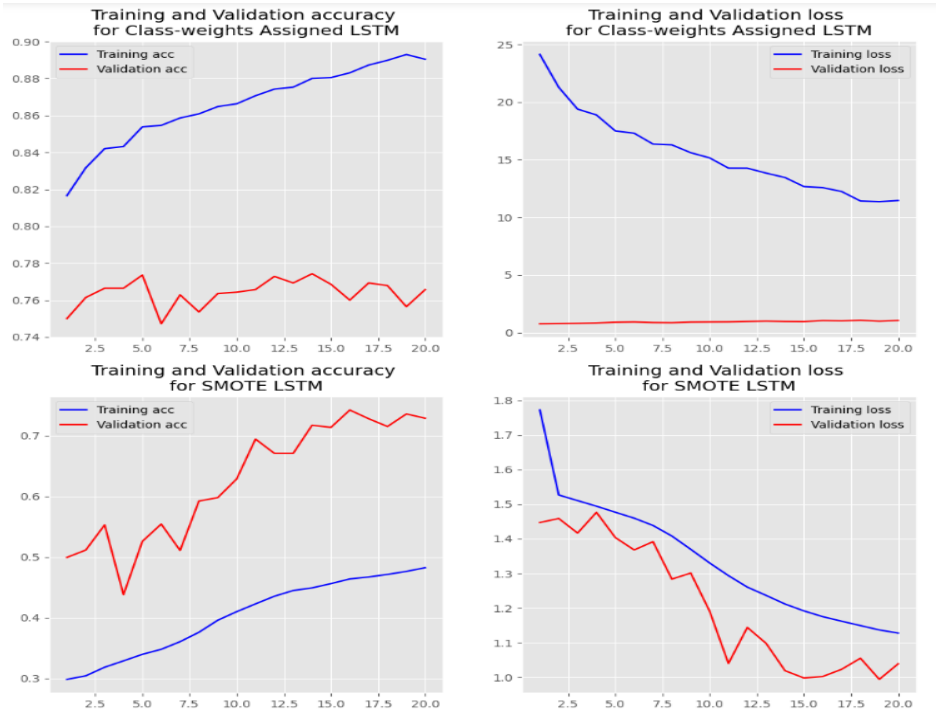


After SMOTE, we are getting the training Accuracy: 0.5126, training Loss: 1.0466, testing Accuracy: 0.7400, testing Loss: 1.0231 and LSTM after SMOTE accuracy score : 0.74.



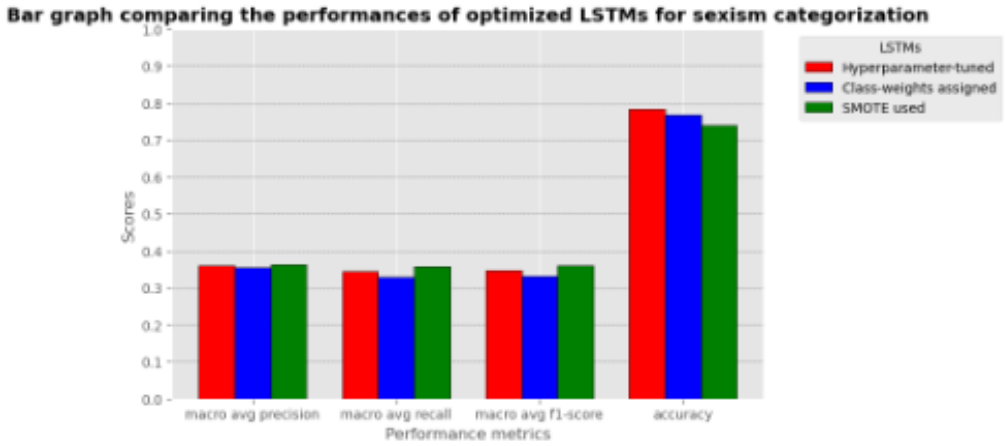
.

Training & Validation loss & accuracy Graph comparing Class-weight & SMOTE:



For LSTM hyperparameter tuned, for the sexism category, the macro avg. F1 score is 0.780. For class weight assignment, for the sexism category, the macro avg. F1 score is 0.77. For SMOTE, for the sexism category, the macro avg. F1 score is 0.74.

Bargraph showing 4 performance metrics:



For binary and categorical classification, the accuracy score is higher as it is an imbalanced dataset and accuracy is not a reliable matrix for dataset because it calculates the sum of the correct prediction/total number of predictions and it does not take into account the incorrect prediction.Which is why, for an imbalance data set we got a higher accuracy. A more reliable matrix for performance evaluation is precision, recall and F1 score which indicated where we failed to make a correct prediction. After imbalance handling, our classes saturation was much more balanced and when our algorithms processed the disbalanced dataset, the accuracy decreased but we noticed that the precision and recall scores improved for some of our classes and the F1 score was more balanced. This indicates that our algorithm did make a little improvement in predicting the classes in a more balanced way. For our categorical classification, our algorithm did not perform well even after imbalance handling as the task was more complex compared to before and extensive hyperparameter tuning and scaling were needed to get improved performance on this categorical classification task comprising of five classes.

References

1. Kirk, H. R., Yin, W., Vidgen, B., & Röttger, P. (2023). SemEval-2023 Task 10: Explainable Detection of Online Sexism. arXiv preprint arXiv:2303.04222.

1. Grosz, D., & Conde-Cespedes, P. (2020). Automatic detection of sexist statements commonly used at the workplace. In Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2020 Workshops, DSFN, GII, BDM, LDRC and LBD, Singapore, May 11–14, 2020, Revised Selected Papers 24 (pp. 104-115). Springer International Publishing.

1. Janakiev, N. (n.d.). *Practical Text Classification With Python and Keras*. Real Python. https://realpython.com/python-keras-text-classification/#convolutional-neural-networks-cnn

1. Yin, W., & Zubiaga, A. (2021). Towards generalisable hate speech detection: a review on obstacles and solutions. PeerJ Computer Science, 7, e598.