Cyberbullying detection using Machine Learning and Natural Language Processing

Himanshu Gupta
hgupta6@ncsu.edu
North Carolina State University

Ashwini Nayak
aunayak@ncsu.edu
North Carolina State University

Sneha Aradhey
svaradhe@ncsu.edu
North Carolina State University

Jayaty
jjayaty@ncsu.edu
North Carolina State University

1 INTRODUCTION AND BACKGROUND

1.1 Problem Statement

It is hard to imagine a world without social media as it has transformed modern day communication from the physical world into cyberspace. However, despite its innumerous benefits, social media has also opened up an avenue for cybercrime, cyber-bullying in particular.

Cyber-bullying is a major public health issue that could inflict psychological, emotional and mental health issues. Hence, there is an urgent need to address this issue to prevent its unacceptable consequences. In this project, we wish to implement machine learning and deep learning techniques to automatically detect instances of cyber-bullying across various social media platforms.

Cyber-bullying is a term used for the utilization of electronic communication to bully a person by the means of an intimidating message of threatening or abusive nature. It refers to the content placed on online platforms which is socially unacceptable - abusive or aggressive words used by a group and individual.[2] Social media platforms like YouTube, Twitter, Facebook are few channels which are exploited by cyberbullies to harass other people.

Our project aims to use Machine Learning algorithms and Natural Language Processing to analyze the text/comments and identify the aggressive texts amongst them.

Keywords: Cyberbullying, Natural Language Processing, Machine Learning, AdaBoost, SVM, Naive Bayes, Bagging Classifier, KNN-Classifier, LSTM, BI-LSTM

1.2 Related Work

This research work is done with the intent of developing a model that can automatically investigate and identify cyber-bullying on social media platforms by exploring related work in this field and building upon it. The papers referred have conducted a similar research.

Muneer et al.[1] used TF-IDF and Word2Vec techniques. TF-IDF considers the expressions of words that are same in all texts. Word2Vec uses two-layers of shallow neural networks. They trained seven machine learning models to classify whether a post on a social media application could be labelled as "cyberbullying" or "non-cyberbullying". The models trained were Linear Regression, Logistic Light Gradient Boosting Machine, Stochastic Gradient Descent, Random Forest, AdaBoost, Multinomial Naive Bayes and Support Vector Machine Classifier.Their Logistic Regression performed better than all other classifiers. Random Forest and AdaBoost had an almost similar accuracy but Random Forest had a better F1 score compared to AdaBoost.[1] Support Vector Machine had the lowest precision and accuracy, however, it also had the best recall score as compared to other classifiers. We referred this paper and implemented Linear regression, SVM and Adaboost algorithms on our dataset, as their analysis gave promising results. Also, these results formed a good baseline for us to compare our results against.

H. Kumar Sharma et al. [2] extracted feature vector sets and stacked them together into a single feature set consisting of count vectors and TF-IDF vectors of both words and characters as tokens with an n-gram sequencing of up to 5 level. Logistic Regression, Support Vector Machine, Random Forest Classifier and Gradient Boosting Machine algorithms were implemented.The authors got good results on their training dataset. We referred this paper as it mostly focused on pattern and feature engineering which was something we wished to explore with our dataset. The features created were made with the intent to separate bullying comments from non-bullying comments. The authors found it difficult to generate impactful features.[2] Also trickling from the problem complexity, features used in one platform performed differently in another. For example, YouTube and Twitter used different features in their own platforms.

Talpur et al. [3] used vectorization to classify different features. They trained five Machine Learning algorithms on the dataset they were using which included Naive Bayes, K-Nearest Neighbours, Decision Trees, Random Forest and Support Vector Machine. The authors were able to compare the performance of different classifiers on Twitter generated data and classify them as none, low, medium, high. Using SMOTE helped balance the data which overall helped slightly improve the model performance as it eliminated the imbalance in data.[3] They observed that overall, Random Forest achieved the best results as compared to the other classifiers that they explored and implemented. We referred this paper as they implemented Random Forest on a similar dataset as ours. From their analysis it was evident that Random Forest was a good model to choose for the data they were working with. As the Random Forest Classifier is similar to the Bagging Classifier, we decided to investigate the Bagging Classifier approach on our dataset.

Sylaja S S et al. [5] used document embeddings generated by Doc2Vec for sequential learning by recurrent neural network architectures as a method for effective classification of comments as cyber aggresive/ non-cyber aggresive on social media. They concluded that Doc2Vec embeddings coupled with an LSTM network gave an a high accuracy. They also found that their LSTM model had outperformed the other models of Simple RNN and GRU. We decided to investigate using neural networks and try utilizing an LSTM model on our dataset.

2 METHOD

2.1 Approach

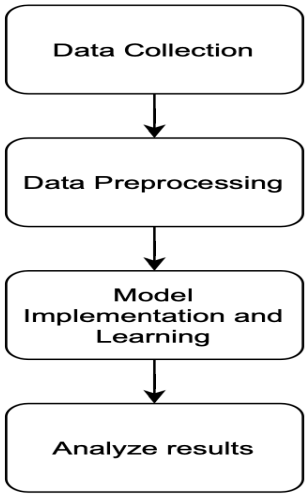

Figure 1: Data Pipeline for the project

We have procured and collected our data from various authentic sources such as Kaggle and Data Mendley. Often unstructured text data consists of noise elements that add little to no value to the context of the sampled data. This could impact the performance of the Machine Learning and Deep Learning models in a negative way as the models try to recognise and learn patterns to account for this noise. This makes pre-processing the data a critical step. As a result, we have catered to this problem by cleaning and pre-processing our data executing the below mentioned steps-

(1) Removal of HTML tags
(2) Expanding Contractions
(3) Vectorization of our text data
(4) Creating Word-Embeddings

The detailed explanation of each of the above pre-processing steps are defined under "Section3.1 - Data".
After pre-processing the data, we chose to implement the following machine learning algorithms to predict whether or not the text represented cyberbullying. The algorithms which we have executed are explained in the following section -

2.1.1 ***AdaBoost.*** Adaptive boosting (AdaBoost) is an ensemble method which is used to boost the performance of decision trees on binary classification problems. It helps to combine multiple "weak classifiers" into a "strong classifier". This is achieved using an iterative approach to learn from the mistakes of weak classifiers, and then convert them into strong classifiers. A weak classifier is one that performs better than random guessing, but still performs poorly at designating classes to objects.
AdaBoost can also be applied on top of any classifier model to learn from its shortcomings and propose a more accurate model.

2.1.2 ***KNN Classifier.*** K-nearest neighbors(KNN) algorithm is a simple, easy to implement and versatile supervised Machine Learning algorithm that can be used to solve both classification and regression predictive problems. KNN is used in a variety of applications such as in building recommendation systems, detecting outliers and searching semantically similar documents where each document is considered to be a vector.
It is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification. It is also a non-parametric learning algorithm as it does not assume anything about the underlying data.
It is used to to predict the target label by finding the class of k-nearest neighbors. The nearest neighbors are found by calculating the Euclidean distance between the two points. Deciding the value of k is one of the most critical part of K-nearest Neighbors. If k is small then noise will have more affect on the result, if k is too large then it could become computationally expensive. Optimal value of k can be found using cross-validation.

2.1.3 ***Support Vector Machine.*** Supervised learning models such as Support Vector Machines are used for two class classification problems such as the one at hand. A hyper-plane or decision boundary is found with the help of algorithms to identify the best separation of the two classes when the data points are plotted on a plane. We select the best hyperparameter as the one whose distance or margin is the closest member is greatest. Where linear data is concerned, classification is simply as defined. Where non-linear data is concerned, the data must be mapped to higher dimensions and this is done with the help of a kernel. A kernel is a function that accepts input in the form of a low dimensional space and maps it to a higher dimensional space.

2.1.4 ***Naive Bayes.*** Naive Bayes Classifier is a probabilistic machine learning model that is used for classification task. It can outperform even highly sophisticated classification methods. It is based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.
The model assumes that each variable is independent, this is a very effective technique on a large range of complex problems. It classifies the data by maximizing P(O|C)P(C) using Bayes Theorem, where O is the tuple in dataset and 'C' is the class instance.

2.1.5 ***Logistic Regression.*** Logistic Regression is a predictive analysis which is useful for binary classification. The algorithm uses the features and predicts based on probability of the class for the input. This algorithm performs better as the size increases. The algorithm iteratively updates the set of parameters and attempts

to minimize the error function. It uses a complex cost function known as the "Sigmoid Function" and maps the real values to values between 0 and 1.
This algorithm can also be extended to multi-class classification, which is then known as Multinomial Logistic Regression.

2.1.6 ***Bagging Classifier.*** A Bagging Classifier is an ensemble method which classifies each on random subset of original data and then aggregates the individual prediction to form a final prediction. Bagging reduces overfitting and strengthens weak classifiers due to the novel ensemble approach it implements.

2.1.7 ***Long Short-Term Memory (LSTM).*** Long Short Term Memory (LSTM) is a Deep Neural Network which remembers values over arbitrary time intervals. LSTM use a gating mechanism that controls the memorizing process which allows the model to store long data in its memory. This is very helpful in identifying the patterns and context in the text data.

2.1.8 ***Bi-Directional Long Short-Term Memory(BI-LSTM).*** Bi-directional Long Short Term Memory (BI-LSTM) is a Deep Neural Network which is similar to LSTM. In BI-LSTM data is feed from forward and backward direction at the same time which allows BI-LSTM to recognize better patterns and context from both the directions. This will help in understanding the context and sentence structure.

Novelty : The ways in our project holds the novel component to it includes:

(1) Natural Language Processing being something we do not cover in class contributes to the novelty of our project.
(2) We have studied and implemented the Vectorization method to convert our text data into vectors by means of complex word vector representations and, additionally have created word embeddings with GloVe.
(3) We successfully executed an Support Vector Machines (SVM) algorithm before the day of lecture on SVM. Considering that SVM is not widely used for the classification problem at hand, we believe it adds novelty to our work.
(4) We have designed and developed NLP neural network models such as LSTM and BI-LSTM which can be considered outside the scope of our syllabus.

2.2 Rationale

We have implemented the following six machine learning models and 2 deep learning models. The models and the reasons for choosing these models are stated below:

(1) **Support Vector Machine (SVM)** : One of the properties of SVM is that it is independent of the dimensionality of the feature space. In text categorization we generally have to work with large number of features(1000), since SVMs are not feature dependent this means we can generalize the model for any unseen text data.
(2) **Naive Bayes** : Naive Bayes works best for text classification scenarios that involve token and vectorization. This is due to its unique probabilistic approach with independent feature contribution.
(3) **Logistic Regression** : Logistic Regression caters well for binary classification problems. It makes no assumptions about distributions of classes in feature space which makes it a good choice for classification problems such as the one at hand.
(4) **Bagging Classifier** : From a statistical point of view, this procedure asymptotically approximates the models sampled from the Bayesian posterior distribution. The model combination strategy for bagging is majority vote. Simple as it is, this strategy can reduce variance when combined with model generation strategies. Previous studies on bagging have shown that it is effective in reducing classification errors thus increasing accuracy.
(5) **AdaBoost** : AdaBoost uses a property of bagging and boosting which promotes incorrectly classified classes. AdaBoost might prove beneficial as it increases the weight of the incorrect classified classes which could result in a better performance.
(6) **KNN Classifier** : KNN is a widely used technique for general classification and pattern recognition problems. KNN is applied for our current problem because of its simplicity. We have converted our text into tokens of vectors to be given as input to the model. KNN algorithm can help find the pattern between those vectors and classify them as the similar instances which we intend to achieve with our results.
(7) **Long Short Term Memory (LSTM)** : LSTMs are a special kind of RNN which are capable of remembering long time dependencies in a sequence. The thought behind using this is to capture basic insights of the text. LSTMs are capable of effectively capturing long term dependencies through its gating mechanism through the concept of layers and gates, thereby tackling the vanishing gradients (a problem we would have faced has we used a simple RNN). A primary feature of LSTM (that is lacking in a GRU network) is the controlled exposure of memory content. Therefore, LSTM networks use the output gate to control the quantity of memory content visible for use by other units in the network. We believe that using an LSTM network may lead to better results to handle large number of data instance.
(8) **Bi-directional LSTM (BI-LSTM)**: BI-LSTM is a slightly advanced version of a regular LSTM model. The main difference between the two is that BI-LSTM will receive inputs in two ways. The first input is from the starting to the end of the sequence and the second input is from the end to the start of the sequence. Due to this extra feature, the model is able to perform better than LSTM as it is able to preserve information both from past and future. Therefore, this approach works well for classification problems as it makes the LSTM more efficient for text data. This is also a result of BI-LSTM having twice the number of parameters than that of a unidirectional LSTM. Figure 2 shows the BI-directional LSTM architecture.

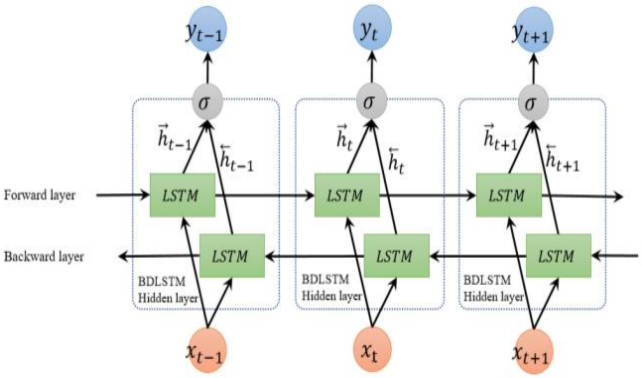

Figure 2: BI-LSTM architecture

3 PLAN AND EXPERIMENTS

3.1 DataSet

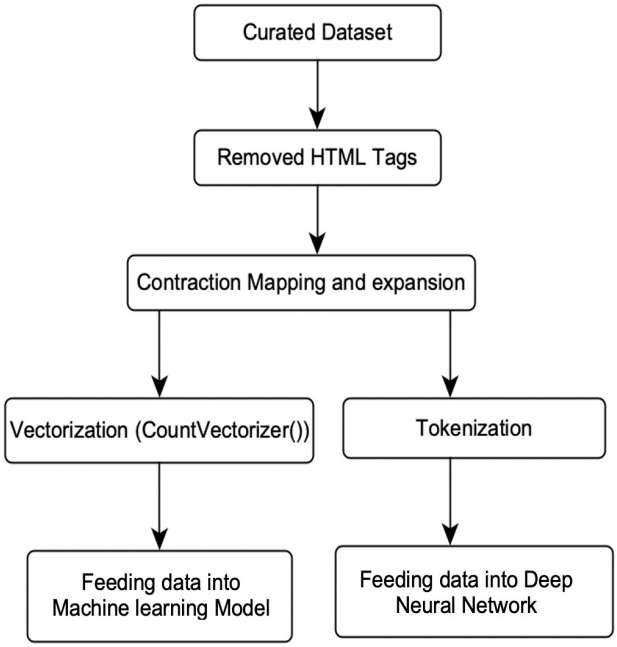

Figure 3: Data Pre-processing Flow

We have curated data from multiple sources related to the automatic detection of cyber-bullying such as Twitter, Kaggle[4] and Youtube. All the datasets have several attributes, two of which are the text and the label which classifies the given text as cyber-bullying or non cyber-bullying.
We have curated our dataset and then transformed the value of the label attribute as 0 or 1.
The label value of 0 or 1 means as follow :
1 - cyber-bullying detected
0 - no cyber-bullying or no hate speech detected

The dataset having comments from Youtube has the following attributes :

- **UserIndex -** The user index rating.
- **Text -** The comment by the user.
- **Number of Subscribers -** Number of subscribers for the user.
- **Membership Duration -** The duration of the membership taken by the user.
- **Age -** Age of the user.
- **Label -** Label classifying the comment as 0 or 1 where 0 is non cyber-bullying and 1 is cyber-bullying.

The dataset having comments from Twitter has the following attributes -

- **Index -** The index to identify a tweet.
- **Id -** The index to identify a tweet.
- **Text -** The tweet by the user.
- **Annotation -** Label classifying the tweet as none or racism or sexism, where none is non cyber-bullying, racism is when cyber-bullying is based on race and sexism is when cyber-bullying is based on sex.
- **Label -** Label classifying the comment as 0 or 1 where 0 is non cyber-bullying and 1 is cyber-bullying.

The dataset collected from Kaggle has the following attributes -

- **Index -** The index to identify a comment.
- **Label -** Label classifying the comment as 0 or 1 where 0 is non cyber-bullying and 1 is cyber-bullying.
- **Date -** The date when user posted a comment.
- **Text -** The comment by the user.

Each dataset has several attributes which are not required for our task in hand such as (Number of Subscribers) and (Age) in Youtube dataset. Hence, we kept only Text and Label attributes from these datasets to maintain the consistency across our datasets and generate our master dataset which we used for our model training and testing. Figure 4 shows the class distribution for our dataset

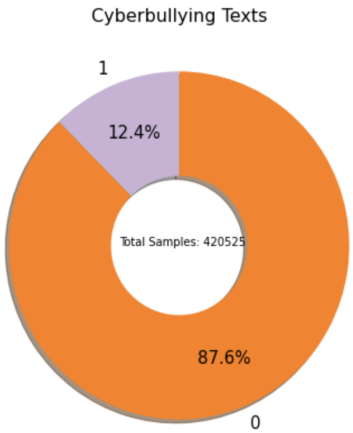

Figure 4: Class Distribution for Dataset

Figure 3 shows the data pre-processing flow for our project. We performed the following pre-processing steps on our data :

(1) **Remove any HTML tags** - Usually the textual data which is collected and scraped from online sources contains some HTML tags which do not add much value to the context of the data. These HTML tags act as noise and can cause unwanted bias in the model. We used the python BeautifulSoup library to strip off any HTML tags from the text.
(2) **Expanding Contractions** - Contractions are the shortened form of English words. For e.g. "he is" can be written as "he's". While performing EDA on our data, we observed there were many contractions in our entire dataset. Hence, we chose to expand all the contractions to the original lengthened word to maintain text standardization through the data set.
(3) **Vectorization of Text data** - The textual data needs to be converted into series of integers or floating point numbers so that it can be fed as input to the machine learning/deep learning models. This process of conversion of textual data to vectors are called vectorization. For our machine learning algorithms, we used Scikit-Learn's library called CountVectorizer() which makes it easy to tokenize the text and will also create a vocabulary of the new words. To implement Neural Networks models, we used Tokenizer() from TensorFlow. Keras library which provides many more features such as word_count, word_index.
(4) **Creating Word-Embeddings** - In word embeddings, all individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network. We are using GloVe (Global Vectors for Word Representation) to create word embeddings.

3.2 Hypothesis

The hypothesis we tried to investigate is to answer the question: "Can we classify a sentence picked up from social media as cyberbullying?" Based on the results of the above hypothesis we will as well try to demonstrate "Which model performs better Deeplearning models or advanced Machine Learning Models for binary text classification?"

3.3 Experimental Design

Our entire project is implemented using Python3. We used multiple python libraries including Pandas, Numpy, Matplotlib and BeautifulSoup for our dataset pre-processing and exploratory data analysis. We used CountVectorizer library from Sklearn and Tokenizer from Keras to perform vectorization of text data. We used GloVe for word embeddings. Additionally, all the machine learning algorithms were imported from the Sklearn library.We used Tensorflow and Keras library and their functions to implement deep learning LSTM and Bi-LSTM model.

Platform : We ran all our experiments on Google Colab notebook as its provides GPU which reduces the machine learning implementation time significantly.

Pre-Processing: We collected the dataset which had text comments from various social media platforms. We combined all the datasets to make our final dataset. We further cleaned our dataset and removed any HTML tags using the BeautifulSoup library in python. It removes any HTML tags which are not useful for our classification task at hand. Furthermore, we created a contraction mapping which contain multiple English words and their contractions. We created a custom function from scratch through which we parsed all our text and it expanded all the contractions using the mapping that we had. We then randomly split our data into a 80% training and 20% test data split.

Vectorization : We used CountVectorizer function from sklearn library converts the text data into vectors. We transformed our training dataset into these vectors using CountVectorizer.fit_transform() function. We also transformed our test dataset to similar vectors using CountVectorizer.transform() function.
After vectorizing our dataset, we then fed it as the input to our training data into machine learning models to train. The machine learning models are imported from Sklearn Library
For deep learning models, we used Tokenizer function from Tensorflow library. We used Tokenizer.fit_on_texts() and Tokenizer.texts_to_sequences() to generate the text vectors.

Word - Embeddings : We used GloVe (Global Vectors forWord Representation) to create word embeddings which is based on matrix factorization techniques on the word-context matrix. After creating the word-embeddings, we passed those embeddings to our deep learning models (LSTM and BI-LSTM).

Traditional Machine learning Algorithms: Since we used CountVectorizer to transform our text into vector each text will be represented as a series of vectors. We fed these vectors to our traditional machine learning algorithms. We checked the performance metrics for all the algorithms which can be seen in Figure 7.

Long Short Term Memory (LSTM): We trained a single layer LSTM model with added Dense and Dropout layers. Figure 5 shows the model summary for our LSTM model.

```
Model: "model_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 30)]              0
_________________________________________________________________
embedding (Embedding)        (None, 30, 300)           91539000
_________________________________________________________________
spatial_dropout1d_7 (Spatial (None, 30, 300)           0
_________________________________________________________________
lstm_9 (LSTM)                (None, 64)                93440
_________________________________________________________________
dense_21 (Dense)             (None, 512)               33280
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_22 (Dense)             (None, 512)               262656
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 513
=================================================================
Total params: 91,928,889
Trainable params: 389,889
Non-trainable params: 91,539,000
_________________________________________________________________
```

Figure 5: LSTM model summary

Bi-Directional LSTM (BI-LSTM: We trained a Bi-directional LSTM model with added Dense and Dropout layers. Figure 6 shows the model summary for our BI-LSTM model.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 30)]              0
_________________________________________________________________
embedding (Embedding)        (None, 30, 300)           91539000
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 30, 300)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               186880
_________________________________________________________________
dense_3 (Dense)              (None, 512)               66048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 92,055,097
Trainable params: 516,097
Non-trainable params: 91,539,000
_________________________________________________________________
```

Figure 6: Bi-Directional LSTM model summary

4 RESULT

4.1 Results

After training and testing different machine learning and deep learning models, we found the values of Precision, Accuracy, F1 Score and Recall score for each model. Accuracy is not a good parameter to compare the models in our case as the data imbalance might cause the model to overfit. In this case we could just be receiving a high accuracy as the model might be biased in predicting the majority class. In other words, we could get a good accuracy score even though the model might not be predicting the desired results. Therefore, the best parameters to compare the models are F1 Score.

Based on the values for these parameters, we found that Naive Bayes, Logistic regression and Bagging Classifier perform better than other models. The Figure 7 shows the metrics comparison between the multiple Machine Learning models.

Model	Accuracy	Precision	Recall	F1 Score
KNN-Classifier	77.4	68.1	31.2	42.8
AdaBoost	92.4	30.2	77.8	43.5
Naïve Bayes	93.6	57.7	70.8	63.6
Logistic Regression	94.6	59.4	79.5	68
Bagging Classifier	94.1	66.6	70.6	68.6
SVM	82.8	83.4	45.6	58.9

Figure 7: Metrics Comparison for Machine Learning Models

Deep learning models (LSTM and BI-LSTM) performs better than machine learning models. Refer to Figure 5 and Figure 6 for summary of our LSTM and BI-LSTM models which we implemented. The various hyperparameters used in the models are -

(1) Epochs - 15
(2) Dropout - 0.2
(3) Optimizer - Adam
(4) Learning Rate - 0.001
(5) Loss Function - Binary Crossentropy
(6) Batch Size - 512

F1 score are comparatively better in Deep Learning Models than Machine Learning models. Figure 8 and Figure 9 show the F1 curve and loss curve correspondingly for LSTM model.

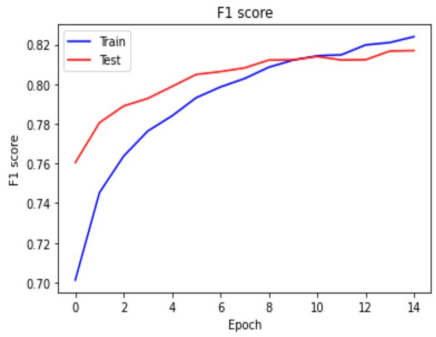

Figure 8: F1 plot for LSTM model

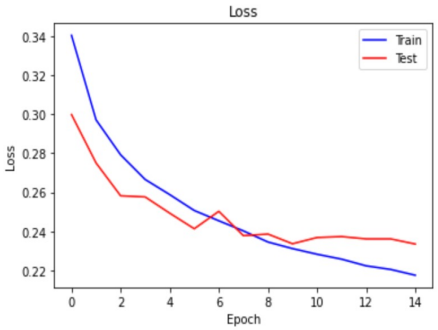

Figure 9: Loss plot for LSTM model

Figure 10 and Figure 11 show the F1 curve and loss curve correspondingly for BI-LSTM model.

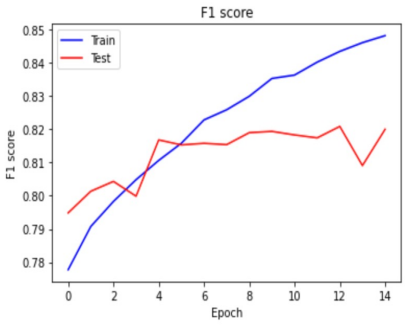

Figure 10: F1 plot for BI-LSTM model

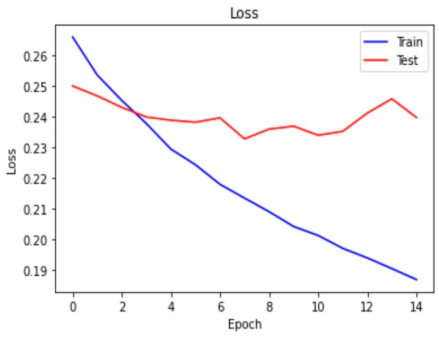

Figure 11: Loss plot for BI-LSTM model

Figure 12 shows the comparison metrics between the performance of our deep learning models.

Model	Accuracy	Precision	Recall	F1-Score
LSTM	89.97	79.61	84	81.6
BI-LSTM	95.7	81.66	82.48	82

Figure 12: Metrics Comparison for Deep Learning Models

4.2 Discussion

As the number of data samples for Class label 0 is more than Class label 1 for our dataset, accuracy might be skewed due to majority class.

Where imbalanced data is concerned, precision and recall are better metrics to gauge performance on as:

- Precision quantifies the number of positive class predictions that actually belong to the positive class.
- Recall quantifies the number of positive class predictions made out of all positive examples in the dataset.
- F1 Score provides a single score that balances both the concerns of precision and recall in one number as it takes the harmonic mean of precision and recall.

Therefore, F1 score is the best metric to determine the performance of our model. We have compared all our Machine learning and Deep Learning models on F1 Score.

(1) *Results for Traditional Machine Learning models*: Figure 7 shows the results of machine learning models with Count Vectorization to create the vectors from the text. As observed from the results, Bagging Classifier and the Logistic Regression perform best than other machine learning models in terms of our measure F1 Score on validation dataset. As we are predicting binary output Logistic regression performs better while Bagging classifier is an ensemble method which is known to work well with skewed dataset. To accurately understand the context, algorithms needs to capture the full sequential information from the text. Since, the traditional ML algorithms are not able to store that, they do not achieve high F1-score as comparison to Deep Learning Models.

(2) *Results for Deep Learning Models*: The Figure 12 summarizes the results for the deep learning models LSTM and BI-LSTM(Refer to Figure 5 and Figure 6 for details on the models). BI-LSTM being a bidirectional 2 layer LSTM is able to capture the textual context better than the LSTM model which can be seen from the result summary, BI-LSTM has better F1 Score. Also, from Fig 8 and Fig 10 we observe that BI-LSTM takes lesser number of epochs i.e. learns faster than LSTM which is expected due to the added advantage of data flowing from both the directions.

5 CONCLUSION

(1) While exploring and analyzing data we realised that it can help to weed out unnecessary information. For our dataset, we saw that keyword attributes UserIndex, Age, Annotation etc. were not present uniformly across all the datasets. As a result, we were able to remove those attributes.

(2) We also saw that the pre-processing step varies with the type of dataset. Majority of the resources that dealt with text pre-processing did not include removal of links and tags ("@"). However, the removal of these tokens was important to the dataset as they served no purpose in classifying text for cyberbullying.

(3) Deep learning models used (LSTM and BI-LSTM) with 64 neurons (>80 F1 Score) are able to learn the context better than the traditional ML algorithms (<70 F1 Score) due to the memory component.

6 MEETING SCHEDULE

We had the following meetings conducted until now for the project. **All of the team members attended all the scheduled meetings and were active throughout to get the work done.**

- 29th-March-2021 - 1:00pm - 2:30pm
- 1st -April-2021 - 1:00pm - 3:00pm
- 3rd -April-2021 - 1:00pm - 3:00pm
- 7th -April-2021 - 1:00pm - 3:00pm
- 11th -April-2021 - 1:00pm - 3:00pm
- 17th -April-2021 - 1:00pm - 3:00pm
- 23rd -April-2021 - 1:00pm - 3:00pm
- 26th -April-2021 - 1:00pm - 3:00pm
- 29th -April-2021 - 1:00pm - 3:00pm

7 CODE

Please refer to the below github repository for our code -
Repo Name: Cyberbullying-Detection
Link: https://github.com/Himanshuu-Gupta/Cyberbullying-Detection

REFERENCES

[1] Muneer, A.; Fati, S.M. A Comparative Analysis of Machine Learning Techniques for Cyberbullying Detection on Twitter. Future Internet 2020, 12, 187.

https://doi.org/10.3390/fi12110187
[2] H. Kumar Sharma, K. Kshitiz and Shailendra, "NLP and Machine Learning Techniques for Detecting Insulting Comments on Social Networking Platforms," 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), Paris, France, 2018, pp. 265-272, doi: 10.1109/ICACCE.2018.8441728.
[3] Talpur BA, O'Sullivan D (2020) Cyberbullying severity detection: A machine learning approach. PLOS ONE 15(10): e0240924. https://doi.org/10.1371/journal.pone.0240924
[4] Kaggle Suspicious Tweet Dataset "www.kaggle.com/syedabbasraza/suspicious-tweets"
[5] Shylaja S S, Abhishek Narayanan, Abhijith Venugopal, Abhishek Prasad, "Recurrent Neural Network Architectures with Trained Document Embeddings for Flagging Cyber-Aggresive Comments on Social Media" ADCOM 2018, Bangalore.