Data Science Intern at Data Glacier

Project: Hate Speech Detection using Transformers (Deep Learning)

Week 13: Deliverables

Name: Shubh Goyal

University: Boston University

Email: Shubhg@bu.edu

Country: USA

Specialization: Data Science

Batch Code: LISUM20

Date: 13th June 2023

Submitted to: Data Glacier

Table of Contents:

1. Project Plan

Week	Plan
Week 07	Problem Statement, Data Collection, Data Report
Week 08	Data Preprocessing (Text Cleaning)
Week 09	Data Preprocessing (Preprocessing Operation + Feature Extraction)
Week 10	Building the Model
Week 11	Model Result Evaluation
Week 12	Flask Development + Heroku
Week 13	Final Submission (Report + Code + Presentation)

2. Problem Statement

Hate speech is defined as any type of verbal, written, or behavioral communication that attacks or uses derogatory or discriminatory language against a person or group because of who they are, such as their religion, ethnicity, nationality, race, color, ancestry, sex, or another identity factor. We will walk you through a hate speech detection model using Machine Learning and Python in this challenge.
Hate Speech Detection is a sentiment categorization job. So, for training, a model that can classify hate speech from a specific piece of text may be produced by training it on sentiment classification data. As a result, we will employ Twitter tweets to identify hate speech for the job of hate speech identification model.

3. Data Collection

The data is about Twitter hate speech acquired from Kaggle [1], and it has 3 characteristics and 31962 observations. It was used to explore hate-speech detection using Twitter data. The material is divided into three categories: hate speech, offensive language, and neither. Because of the study's nature, it is necessary to mention that this dataset contains content that might be deemed racist, sexist, homophobic, or objectionable.

Total number of observations	31962
Total number of files	1
Total number of features	3
Base format of the file	CSV
Size of the data	2.95

4. Data Preprocessing

4.1 Text Cleaning

First, we clean our text because it was so messy data.

- 4.1.1 Lowercase

Converting a word to lower case (NLP -> nlp). Words like Racism and racism mean the same but when not converted to the lower case those two are represented as two different words in the vector space model (resulting in more dimensions). Therefore, we convert all text word into lower case letter.

- 4.1.2 Remove Punctuation

It is important to remove the Punctuation because is not important. Therefore, we remove that. Punctuation to do that we use regular expression.

- 4.1.3 Remove URLs

In this part, we remove URLs because we are working on hate speech application which detect the hate and free speech and to get the output, we need to give only text not URLs therefore, we remove the URLs because we need only clean text input.

- 4.1.4 Remove @tags

In this part, we remove @tags which basically used when we mentioned someone So, it's doesn't concern to our application therefore, we remove @tags by using regular expressions.

- 4.1.5 Remove Special Characters

Remove Special Characters is essentially the following set of symbols [!"#$%&'()*+,-./:;<=>?@[]^_`{|}~] which basically don't have meaning. Therefore, we remove that kind of symbols because we don't need that. To remove we use python isalnum method.

4.2 Preprocessing Operation

In this part, we implement the preprocessing operation

- 4.2.1 Tokenization

Tokenization is breaking the raw text into small chunks. Our text data is into paragraph so to convert into work tokenize we use nltk work_tokenize library. These tokens help in 5 understanding the context or developing the model for the NLP. The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

- 4.2.2 Removing StopWords

  StopWords is basically 'a,' 'is,' 'the,' 'are' etc. If we see our dictionary, then these words do not have meaning and don't need that to build Hate speech detection application. To remove stop words from a sentence, we divide text into words which we did above in tokenization and then remove the word if it exits in the list of stop words provided by NLTK. To do that, we first import the StopWords collection from the nltk.

- 4.2.3 Lemmatization

  Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is like stemming but it brings context to the words. So, it links words with similar meanings to one word. Like the word Intelligently, intelligence, convert into root form intelligent.

- 4.2.4 WordCloud

  A Worldcloud is a visual representation of text data, which is often used to depict keyword metadata on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color.

4.3 Feature Extraction

- 4.3.1 TF-IDF Model

  Once the dictionary is ready, we apply Term Frequency-Inverse Document Frequency (TFIDF) model, and we take 2000 most frequent words from dictionaries for each Hate/Free Speech of the whole dataset. Each word count vector contains the frequency of 2000 words in the whole dataset file.

4.4 Split the Data into Train into Test

In this part, we split the data into Train. And we split 80% for training and 20% for test. Data splitting is when data is divided into two or more subsets. Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model. Data splitting is an important aspect of data science, particularly for creating models based on data.

4.5 Build the Model

In this part, we build the CNN with LSTM Model using TensorFlow.

- 4.5.1 CNN with LSTM

The CNN LSTM architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction. This architecture was originally referred to as a Long-term Recurrent Convolutional Network or LRCN model, although we will use the more generic name "CNN LSTM" to refer to LSTMs that use a CNN as a front end in this lesson.

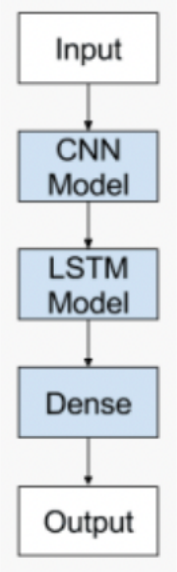


Figure 2: CNN with LSTM Model [3]

5. Result Evaluation and Discussion

In this chapter, we evaluate our Result and define the evaluation criteria to calculate the performances of our best classification model.

5.1 Evaluation Criteria

The confusion matrix was used to evaluate the classification models throughout the training process. The confusion matrix is a table that compares predicted and actual outcomes. It is frequently used to describe a classification model's performance on a set of test data.

Class	Predicted Negative	Predicted Positive
Actual Negative	TN	TP
Actual Positive	FN	FP

Important metrics were constructed from the confusion matrix in order to evaluate the classification models. In addition to the accurate classification rate or accuracy, other metrics for evaluation included True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Negative Rate (FNR), Precision, F1 score, and Misclassification rate.

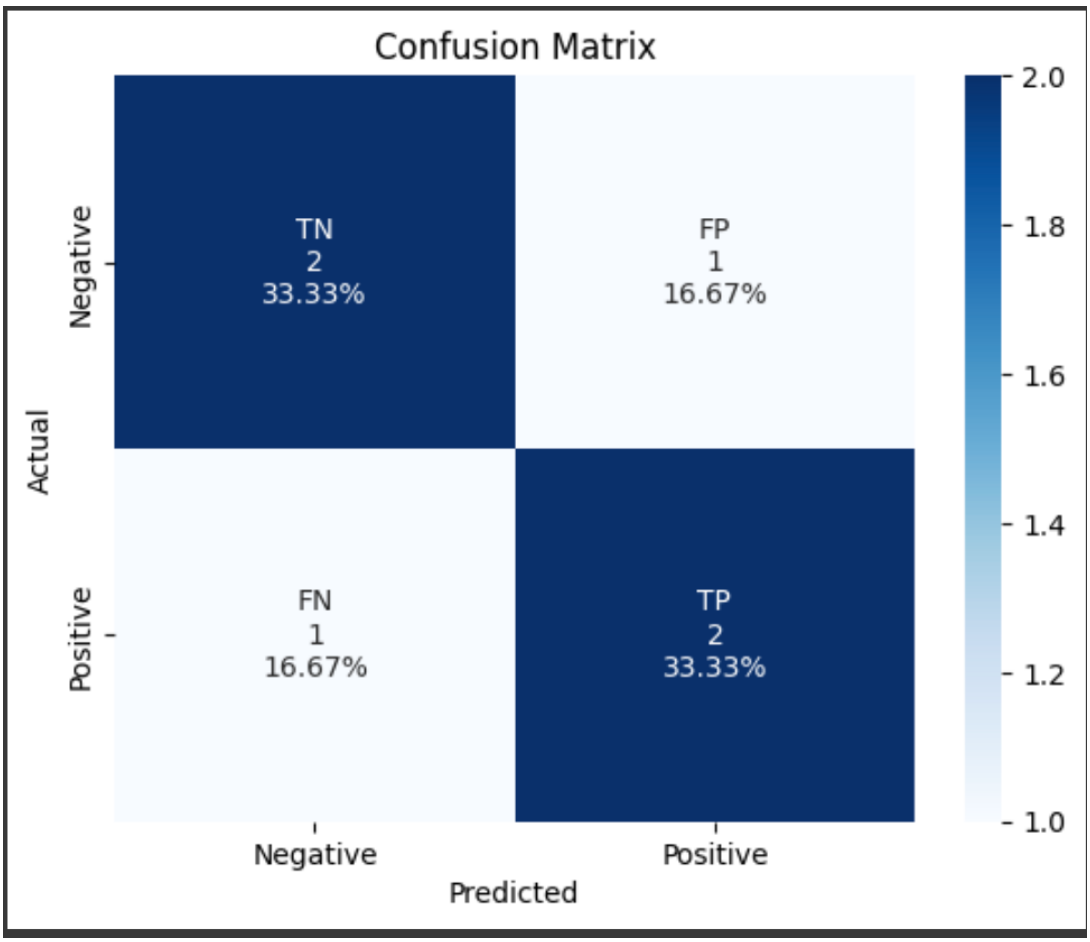
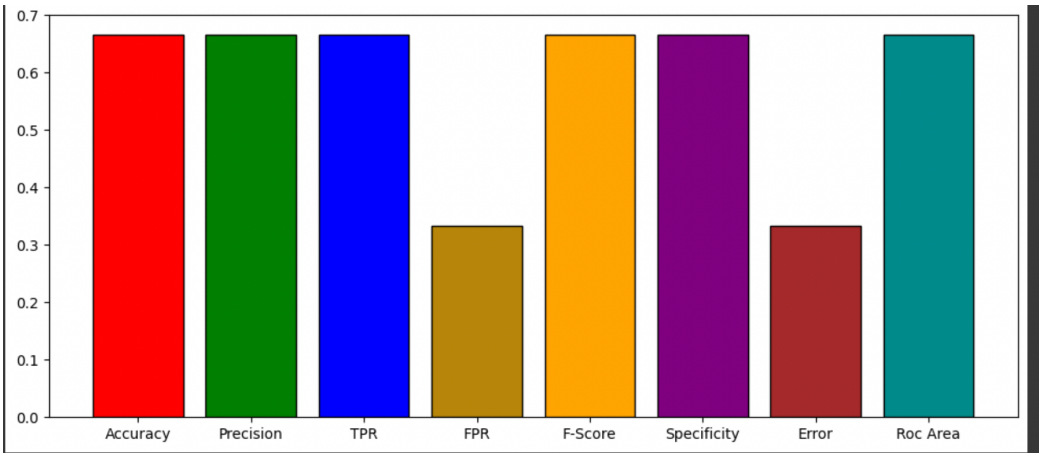


Table shows the final result that we evaluate on the basis of confusion matrix result

Classifiers	Accuracy	Precision	TPR	FPR	F1 Score	Error Rate	Specificity
CNN with LSTM	0.9577	0.8382	0.4191	0.0055	0.5588	0.0422	0.9944



6. Conclusion

The goal of this project was to find capable methods and settings that could be used to help the detection of Hate and Free Speech of twitter. The error rate of the model is not zero, so still, some incorrect can be classified as true by the model. In future we will enhance this work by implementing Temporal Convolutional Network (TCN) and Random Multimodel Deep Learning (RMDL) Techniques