**DL-AS2-REPORT**

**Naveen Kumar Pathi 500187816**

**Venkatesh Policherla 500194692**

**Sai Chand Devarapalli 500192020**

**Krishna Vamsi Vanga 500187921**

**Dorai Charan Muthineni 500185125**

**Problem Statement: -** In this project, we try to implement a Twitter sentiment analysis model that helps to detect the negative tweets.

Objective: -To use deep learning models like Simple-RNN on a NLP project to detect the negative tweets.

**About Dataset: -** The tweets have been pulled from Twitter and manual tagging has been done to them.

<https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification?select=Corona_NLP_train.csv>

The names and usernames have been given codes to avoid any privacy concerns.

A screenshot of a computer

Description automatically generated

**Data Cleaning: -**

Here we keep only ‘Original Tweet’ and ‘Sentiment’ columns and remove the rest. In the sentiment column we replace ‘Negative’ and ‘Extremely Negative’ with 1 and the remaining to 0.

We see that there are no null values present in the dataset. Though our data is imbalanced we need no not make any changes as the deep learning models we use do not require any. Cleaned data looks as below.

A screenshot of a computer

Description automatically generated with medium confidence

**Tokenization: -**

Tokenization is breaking the raw text into small chunks. Tokenization breaks the raw text into words, sentences called tokens.

In an RNN we input a sentence word by word. We represent every word as one hot vectors of dimensions: Numbers of words in Vocab +1.

What keras Tokenizer does is, it takes all the unique words in the corpus, forms a dictionary with words as keys and their number of occurrences as values, it then sorts the dictionary in descending order of counts. It then assigns the first value 1, second value 2 and so on. So, let's suppose word 'the' occurred the most in the corpus then it will assign index 1 and vector representing 'the' would be a one-hot vector with value 1 at position 1 and rest zeros.

A screenshot of a computer

Description automatically generated with medium confidence

**Model: Simple-RNN**

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer.

Text

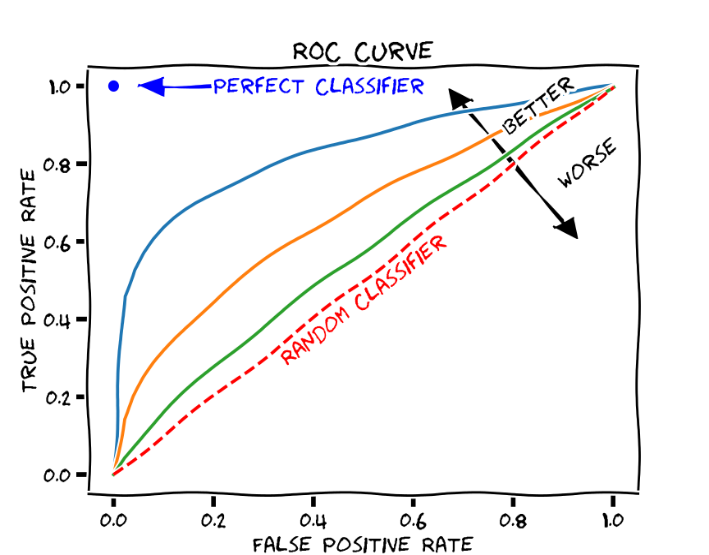
Description automatically generated

**Fitting the model:**

Text

Description automatically generated

AUC Score: Area Under Curve (AUC) score **represents the degree or measure of separability**. A model with higher AUC is better at predicting True Positives and True Negatives. AUC score measures the total area underneath the ROC curve. AUC is scale invariant and threshold invariant.



**Final Observations: - We found AUC score for simple RNN model is 0.83.**

Text

Description automatically generated

**Conclusion: -** Hence, we have successfully used a deep learning model- simple RNN on an NLP project with an AUC score of 0.83.