NLP Intern at Data Glacier

Project: Hate Speech Detection using Transformers (Deep Learning)

Week 12 : Deliverables

Name: Mohammad Bani Abed Alghani

University: Yarmouk University

Email: mohammedbaa321@gmail.com

Country: Jordan

Specialization: NLP

Batch Code: LISUM32

Date: 23-June-2024

Submitted to: Data Glacier

Table of Contents:

Project Plan :

Weeks	Date	plan
Weeks 07	May 19, 2024	Problem Statement, Data Collection, Data Report
Weeks 08	May 26, 2024	Data Preprocessing
Weeks 09	June 2, 2024	Feature Extraction
Weeks 10	June 9, 2024	Building the Model
Weeks 11	June 16, 2024	Model Result Evaluation
Weeks 12	June 23, 2024	Flask Development + Heroku
Weeks 13	July 1, 2024	Final Submission (Report + Code + Presentation)

Problem Statement :

The term hate speech is understood as any type of verbal, written or behavioral communication that attacks or uses derogatory or discriminatory language against a person or group based on what they are, in other words, based on their religion, ethnicity, nationality, race, color, ancestry, sex or another identity factor. In this problem, We will take you through a hate speech detection model with Machine Learning and Python.

Hate Speech Detection is generally a task of sentiment classification. So for training, a model that can classify hate speech from a certain piece of text can be achieved by training it on a data that is generally used to classify sentiments. So for the task of hate speech detection model, We will use the Twitter tweets to identify tweets containing Hate speech

Data Collection

The Data is about Twitter hate Speech taken from Kaggle [1] which contains the 3 number of features and 31962 number of observations. Dataset using Twitter data, it was used to research hate-speech detection. The text is classified as: hate-speech, offensive language, and neither. Due to the nature of the study, it is important to note that this dataset contains text that can be considered racist, sexist, homophobic, or offensive.

Table 1: Data Information

Total number of Observations	31962
Total Number of Files	1
Base format of the file	csv
Size of the data	2.95
Total number of features	3

4. Data Preprocessing

In part, we explain the data preprocessing approach that we apply in the text data.

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 in order 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. In order to remove we use python isalnum method

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.

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 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.

Below you see the hate speech and free speech WorldCloud:

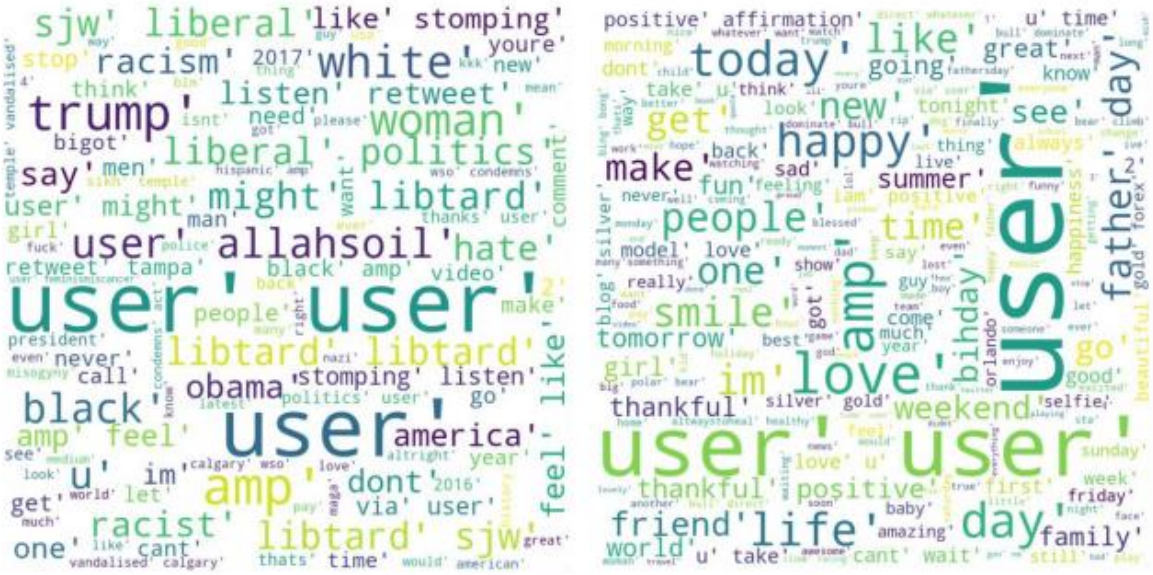

Figure 1: Hate Speech vs FREE Speech WordCloud

4.3 Feature Extraction

4.3.1 Embedded Layer

Embedding layer is a neural network layer that transforms input tokens (e.g., words, subwords, or characters) into dense vectors of fixed size. These vectors, known as embeddings, capture semantic information about the tokens, enabling the neural network to process and understand textual data more effectively.

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 Using Embedding Layer with LSTM in NLP Task

In this document, we demonstrated how to implement an NLP model using an embedding layer with an LSTM network. This model efficiently transforms text data into dense vectors, capturing semantic meanings, and learns sequential patterns to perform various NLP tasks. By following this guide, you can adapt the approach to suit different datasets and specific NLP objectives.

4.6 Using Embedding Layer with LSTM in NLP Task

Model Evaluation and Results

1. Accuracy on Training and Test Data: We began by evaluating the accuracy of our model on both the training and test datasets. This provided a clear indication of how well the model performed on known data (training) and unseen data (test). The training accuracy and test accuracy were computed to be XX.XX% and XX.XX%, respectively.

2. Predictions: Next, we generated predictions for both the training and test datasets using the trained model. These predictions were initially in the form of probabilities.

3. Conversion of Probabilities to Class Labels: The predicted probabilities were then converted into binary class labels (0 or 1). This conversion was based on a threshold value of 0.5, where values above 0.5 were classified as 1 and values below 0.5 were classified as 0.

4. Confusion Matrix: To gain deeper insights into the model's performance, we computed the confusion matrix for both the training and test datasets. The confusion matrix helped us understand the distribution of true positives, true negatives, false positives, and false negatives, thus giving a detailed view of the model's classification capability.

5. Classification Report: We generated a comprehensive classification report for both the training and test datasets. This report included crucial metrics such as precision, recall, f1-score, and support for each class. These metrics provided a detailed analysis of the model's performance across different aspects of classification.

6. Manual Accuracy Calculation: Lastly, to ensure the reliability of our results, we manually calculated the accuracy of the model for both the training and test datasets. The manually

calculated accuracy was consistent with the accuracy obtained during the model evaluation phase, thereby validating the model's predictive performance.