



Data Science Intern at Data Glacier

Project: Hate Speech Detection using Transformers (Deep Learning)

Week 11: Deliverables

Name: Anitha Venkatachalam

Email: anitha.Venkat83@gmail.com

Country: Canada

Specialization: Data

Science **Batch Code:**

LISUM14

Date: 22Nov 2022

Submitted to: Data Glacier

Table of Contents:

1. Project Plan	3
2. Problem Statement	3
3. Data Collection	3
4. Data Preprocessing	4
4.1. Text Cleaning	4
4.1.1. Lower Case	4
4.1.2. Remove Punctuation	4
4.1.3. Remove URL	4
4.1.4. Remove @tags	4
4.1.5. Remove Special Characters	4
4.2. Preprocessing Operations	4
4.2.1. Word Cloud	4
4.3. Feature Extraction	5
4.3.1. TF-IDF Model	5
4.4. Split the Dataset into Train and Test	6
5. Build the Model	6
5.2.1 Logistic Regression with TF-IDF on N-Grams	6
6. Result Evaluation	6
6.1 Evaluation Criteria	6
7. Application Design	7
7.1. Turning ML model Into Flask Framework	8
7.2. Running Procedure	9
8. Conclusion	10

Project Plan

Weeks	Date	plan
Weeks 07	Nov 14th, 2022	Problem Statement, Data Collection, Data Report
Weeks 08	Nov 21, 2022	Data Preprocessing (Text Cleaning)
Weeks 09	Nov 28, 2022	Data Preprocessing (Preprocessing Operation + Feature Extraction)
Weeks 10	Dec 5, 2022	Building the Model
Weeks 11	Dec 12, 2022	Model Result Evaluation
Weeks 12	Dec 19, 2022	Flask Development + Heroku
Weeks 13	Dec 26, 2022	Final Submission (Report + Code + Presentation)

1. 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 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 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.

2. 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
Total number of features	3
Base format of the file	csv
Size of the data	2.95 MB

3. Data Preprocessing

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

3.1 Text Cleaning

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

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.1.4 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.

3.2 Preprocessing Operation

In this part, we implement the preprocessing operation

3.2.1 Word Cloud

A Word cloud 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.

3.3 Feature Extraction

3.3.1 TF-IDF Model

Once the dictionary is ready, we apply Term Frequency-Inverse Document Frequency (TF-IDF) 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.

3.3.2 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.

5.2 Build the Model

5.2.2 Logistic Regression with TF-IDF on N-Grams

This process explains how to run a classification algorithm and more specifically a logistic regression of a “The hate and free speech detection on twitter” using as features the TD-IDF of unigrams, bi-grams, and trigrams. We can easily apply any classification, like Random Forest, Support Vector Machines etc. Finally, it finds whether the text is hated speech or free speech. logistic regression:

Logistic regression is a supervised machine learning method which is like linear regression but instead of using a linear equation it uses a sigmoid function which makes the output value in specific range which is used for text classification also Hate Speech Detection in Twitter using Natural Language Processing.

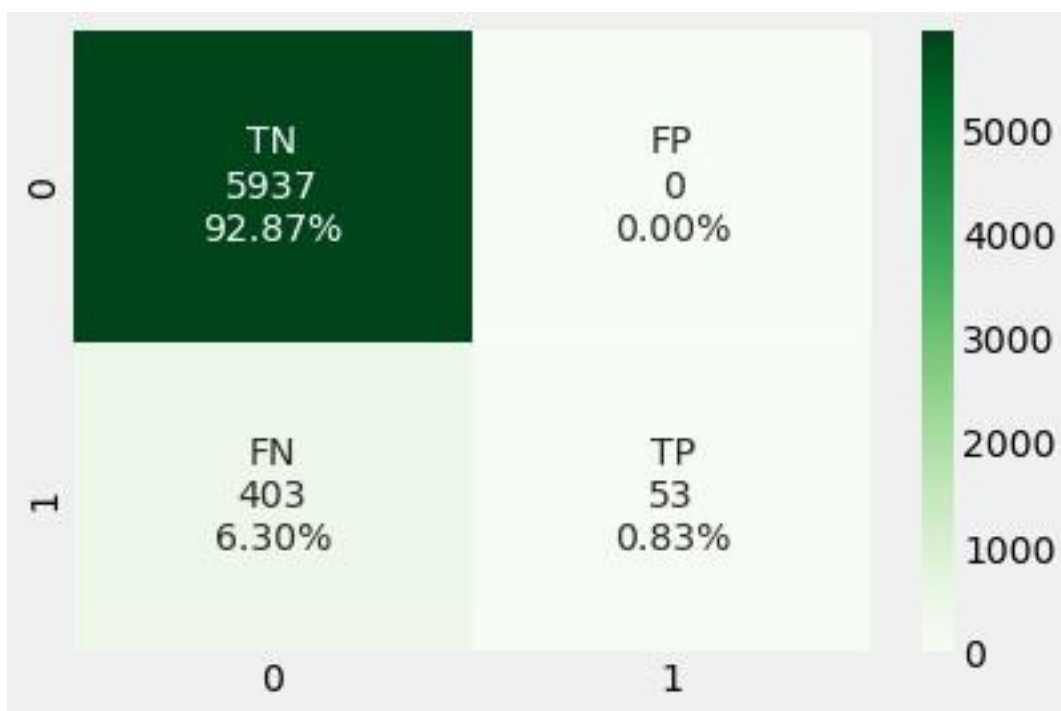
6.1 Evaluation Criteria

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.

Table 1: Confusion Matrix

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

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.

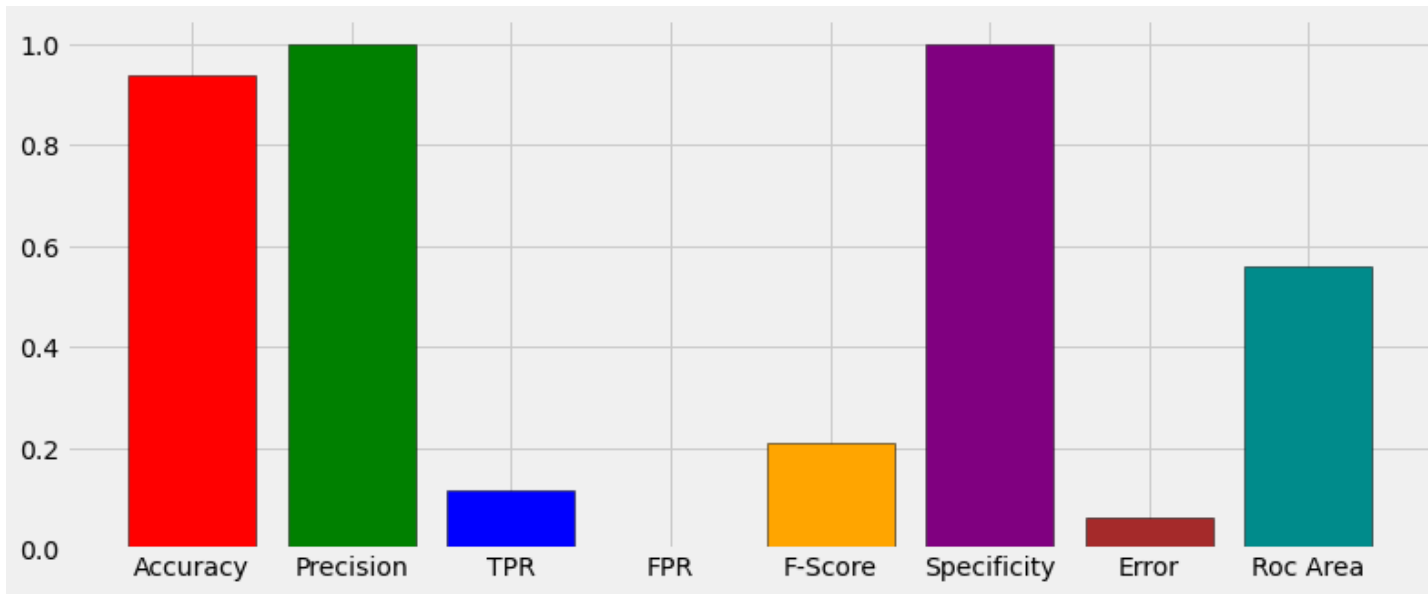


Below table shows the result that we evaluate based on confusion matrix result

Table : Results

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

Below you can see the visualization result of above table as well.



5. Application Design

In this chapter, we develop a Model & Deploy It with Flask. Our model systems workflow is likethis: Train offline -> Make model available as a service -> Predict online.

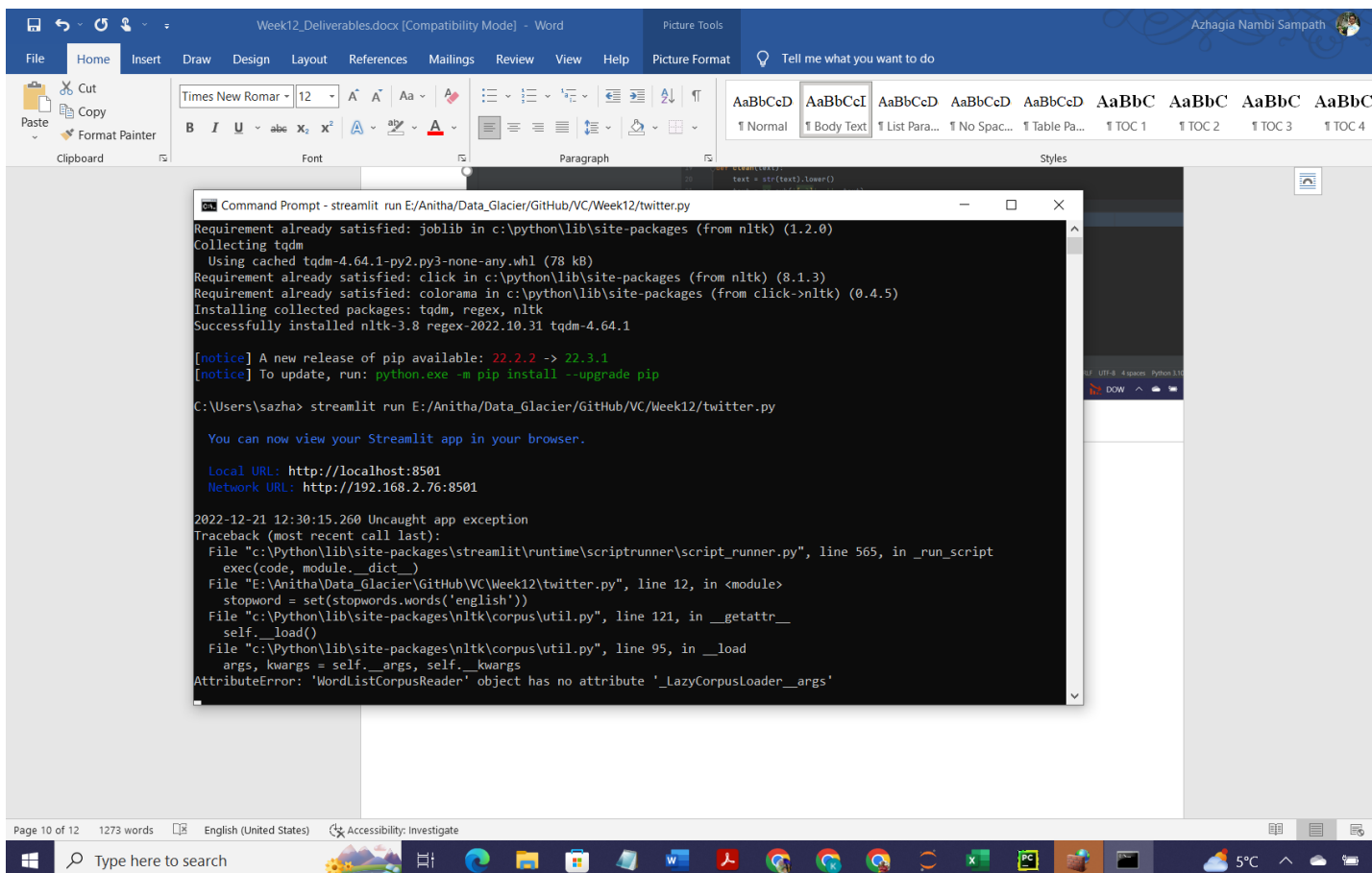
- A classifier is trained offline with Fake and True news.
- The trained model is deployed as a service to serve users.

5.1 Turning Model into a Web application:

First, we use the twitter hate speech dataset to build a prediction model that will accurately classify hate speech. We develop a web application that consists of a simple web page with a form field that lets us enter a message. After submitting the message to the web application, it will render it on a new page which gives us the result of hate speech/free speech.

5.2. Running Procedure

Once we have done all of the above, we can start running the API by either double click `app.py`, or executing the command from the Terminal:



```
Week12_Deliverables.docx [Compatibility Mode] - Word
File Home Insert Draw Design Layout References Mailings Review View Help Picture Tools
Tell me what you want to do
Clipboard Font Paragraph Styles
Times New Roman 12 A A A
B I U abc X x
AaBbCcD AaBbCcI AaBbCcD AaBbCcD AaBbCcD AaBbC AaBbC AaBbC AaBbC
1 Normal 1 Body Text 1 List Para... 1 No Spac... 1 Table Pa... 1 TOC 1 1 TOC 2 1 TOC 3 1 TOC 4

Command Prompt - streamlit run E:/Anitha/Data_Glacier/GitHub/VC/Week12/twitter.py
Requirement already satisfied: joblib in c:\python\lib\site-packages (from nltk) (1.2.0)
Collecting tqdm
  Using cached tqdm-4.64.1-py2.py3-none-any.whl (78 kB)
Requirement already satisfied: click in c:\python\lib\site-packages (from nltk) (8.1.3)
Requirement already satisfied: colorama in c:\python\lib\site-packages (from click->nltk) (0.4.5)
Installing collected packages: tqdm, regex, nltk
Successfully installed nltk-3.8 regex-2022.10.31 tqdm-4.64.1

[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

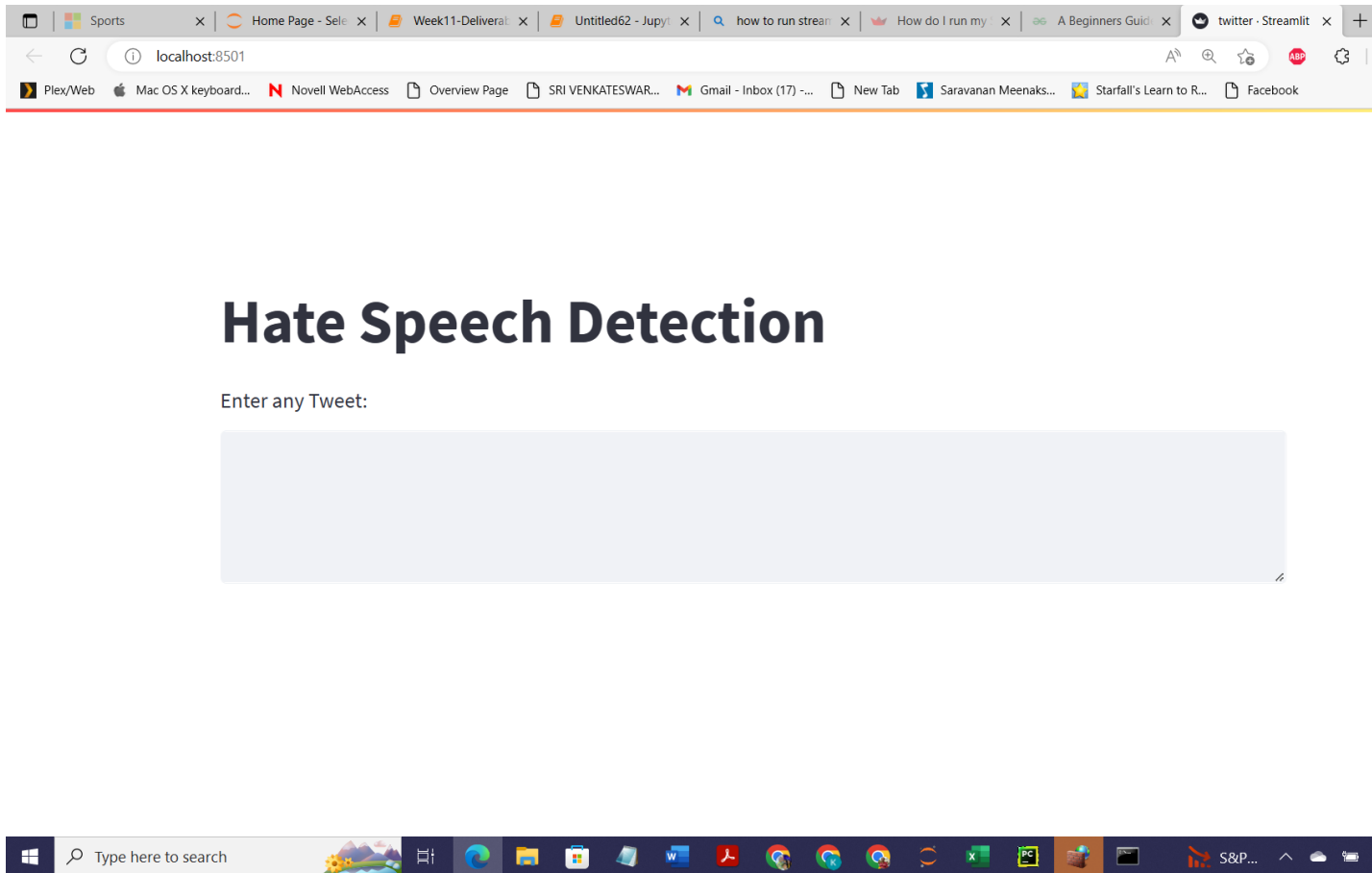
C:\Users\sazha> streamlit run E:/Anitha/Data_Glacier/GitHub/VC/Week12/twitter.py

You can now view your Streamlit app in your browser.

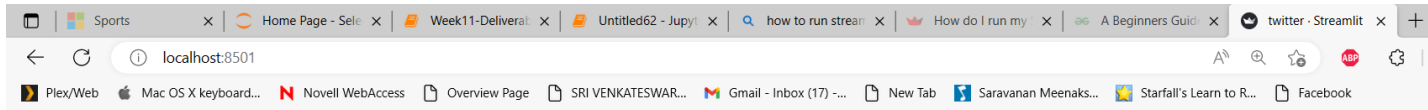
Local URL: http://localhost:8501
Network URL: http://192.168.2.76:8501

2022-12-21 12:30:15.260 Uncaught app exception
Traceback (most recent call last):
  File "c:\Python\lib\site-packages\streamlit\runtime\scriptrunner\script_runner.py", line 565, in _run_script
    exec(code, module.__dict__)
  File "E:\Anitha\Data_Glacier\GitHub\VC\Week12\twitter.py", line 12, in <module>
    stopwords = set(stopwords.words('english'))
  File "c:\Python\lib\site-packages\nltk\corpus\util.py", line 121, in __getattr__
    self.__load()
  File "c:\Python\lib\site-packages\nltk\corpus\util.py", line 95, in __load
    args, kwargs = self.__args, self.__kwargs
AttributeError: 'WordListCorpusReader' object has no attribute '_LazyCorpusLoader__args'
```

Now we could open a web browser and navigate to <http://localhost:8501> we should see a simple website with the content like so



Now we enter input in the comments form



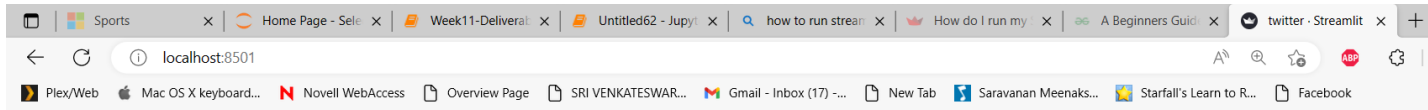
Hate Speech Detection

Enter any Tweet:

🔗 ['No Hate and Offensive Speech']



After entering the input click the predict button now we can the result of our input.



Hate Speech Detection

Enter any Tweet:

how are you?

['Offensive Speech']



7. 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 Multimodal Deep Learning (RMDL) Techniques.

