Project: Hate Speech detection using Transformers (Deep Learning)

Project Report

- **Group name:** Speechium
- **Names:** Richard Flores, Christos Christoforou
- **Email:** rflo147@wgu.edu, christoschristoforou97@hotmail.com
- **Country:** United States, United Kingdom
- **Specialization:** NLP

- **GitHub repo link:** https://github.com/DataDaimon/twitter-sentiment

Problem Statement:

The term hate speech is understood as any type of verbal, written or behavioural 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, colour, 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.

Business Understanding

Detection of hate speech in tweets is an important issue for businesses to consider for several reasons.

First, hate speech can be harmful and offensive to individuals and groups, and businesses have a social responsibility to address it. In addition, businesses may face legal and reputational risks if they fail to address hate speech on their platforms.

Second, businesses that operate social media platforms or engage in social media marketing may need to monitor and address hate speech to maintain the trust and loyalty of their users and customers. If a business is perceived as tolerating hate speech, it may face backlash from users and negative media attention.

Finally, businesses may also have a financial incentive to address hate speech, as it can negatively impact the user experience and drive users away from the platform.

To detect hate speech in tweets, businesses may use a combination of automated tools and human moderation. Automated tools may include machine learning algorithms that are trained to identify hate speech based on certain characteristics, such as the use of certain words or phrases. Human moderation may involve a team of moderators who review tweets and take appropriate action, such as deleting the tweet or banning the user.

It's important to note that detecting hate speech can be challenging, as it may involve complex issues of context and intent. It is also important for businesses to consider the potential for false positives and ensure that their approaches to detecting and addressing hate speech are fair and transparent.

Project lifecycle and deadlines:

- **Week 7:** Data collection & Problem description | **Due date:** 19/12/2022
- **Week 8:** Data understanding & preprocessing | **Due date:** 26/12/2022
- **Week 9:** Data cleansing and transformation | **Due date:** 02/01/2023
- **Week 10:** Model Building & Training | **Due date:** 09/01/2023
- **Week 11:** Model Evaluation & Selection | **Due date:** 16/01/2023
- **Week 12:** Model Deployment | **Due date:** 23/01/2023
- **Week 13:** Final Report, Code, and Presentation | **Due date:** 30/01/2023

Data Intake Report:

Name: Hate Speech Detection

Report date: 29/01/2023

Internship Batch: LISUM15

Version: 1.0

Data intake by: Richard Flores, Christos Christoforou

Data storage location: https://www.kaggle.com/vkrahul/twitter-hate-speech?select=train_E6oV3lV.csv

Tabular data details: train_E6oV3lV

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	3MB

Tabular data details: test_tweets_anuFYb8

Total number of observations	17197
Total number of files	1
Total number of features	2
Base format of the file	.csv
Size of the data	1.6MB

Data understanding

In this section, we familiarise ourselves with the data and try to understand them. In our case, we have 3 features in the training dataset, which are pretty much straight forward to understand. Namely, we have:

- `id': the primary key,
- `label': 0 for free speech, 1 for hate speech,
- `tweet': the tweet we want to classify.

Since the `id' column is not needed for our purpose, we will drop it. Next, we checked the shape of the dataset, the type of each feature, and if there are any null values or duplicated rows.

- Feature types were correct, `label' : int64 and `tweet': object,
- No null-values,
- Found and removed 2432 duplicated rows.

Finally, we checked if our dataset is balanced. In other words, whether we have about the same number or free speech and hate speech tweets or not. We found out that our dataset is highly imbalanced. To address this issue, we will later apply an oversampling or a downsampling technique to balance our dataset.

Data cleansing and transformation completed:

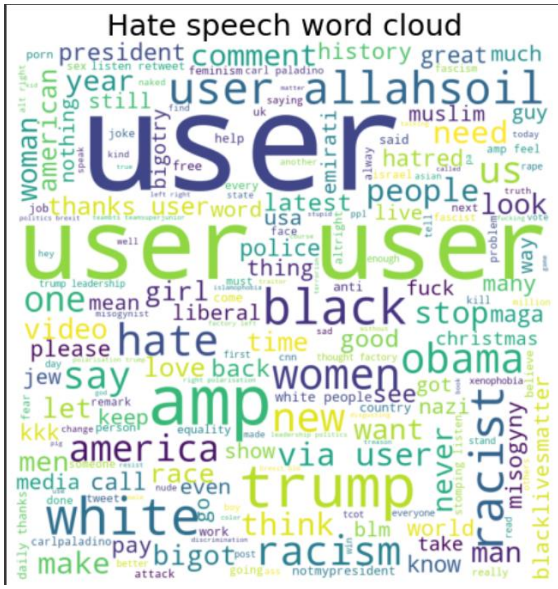
- **Standard Data Cleaning and Transformation**
    - Verify data types
    - Check for NULL values
    - Remove duplicated data
    - Check for missing data

- **NLP Specific Data Cleaning and Transformation**
    - Apply Tokenization and Lemmatization
    - Lowercase the text
    - Remove stop-words and one-letter words
    - Remove tags and other special characters

- Remove non-ASCII characters
- Vectorize the training data using the TfidfVectorizer

EDA – WordCloud

A word cloud is a visualization of the frequency of words in a document. It's typically a graphical representation of the words in a text or passage, with the size of each word indicating its frequency or importance. Word clouds are often used to quickly provide an overview of the key themes or ideas in a document, and can be an effective tool for text analysis and data visualization. We created two word clouds, one for Hate Speech and one for Free Speech.



Splitting the data & Feature extraction

After splitting the data into two sets, training and testing, we apply the TF-IDF (term frequency-inverse document frequency) vectorization. TF-IDF vectorization is a process that converts a collection of documents into a matrix of TF-IDF features, where each document is represented as a row and each word in the vocabulary is represented as a column. The value in the cell at the intersection of a row (document) and a column (word) is the TF-IDF score for that word in that document. The resulting matrix of TF-IDF features can be used as input to a model, and the model can learn to make predictions based on the importance of different words in the documents.

Model Building & Training

- First, we built a **simple transformer model** and train it using our training dataset. We used the Adam optimizer, 5 epochs, and the Cross-Entropy loss as the loss function.

- Next, we fine-tuned the **pretrained Roberta-base model** from the 'simpletransformers' library. We used the default optimizer and loss function, which are the Adam optimiser and Cross-Entropy loss function respectively.

We saved our models to later evaluate them on new unseen data.

Model Evaluation & Selection

We will use a range of scoring metrics to evaluate our models. Some of them are: Precision, Recall, and F1-score.

Here are the formulas for each one:

- recall = TP / (TP + FN)
- precision = TP / (TP + FP)
- F1-score = 2 * ((precision * recall) / (precision + recall)),

where TP, TN, FP, and FN are the True Positives, True Negatives, False Positives, and False Negatives respectively. The table below, also known as a confusion matrix, explains each one of them.

True Class	Predicted Positive	Predicted Negative
Positive	TP	FN

Negative	FP	TN

- **Performance of the simple transformer model:**

	precision	recall	f1-score
0	0.67	0.47	0.56
1	0.29	0.49	0.37
accuracy			0.48

Based on the classification report above, the simple transformer model performs poorly.

- **Performance of the fine-tuned pretrained Roberta-base model from the 'simpletransformers' library:**

Recall: 0.91	Precision: 0.89
Accuracy: 0.9	F1-score: 0.9
AUROC: 0.95	AUPRC: 0.96

This second model seems to perform quite good with scoring metrics around 90%.

At this point, it is obvious that we will continue with the fine-tuned Roberta-base model for making predictions on unseen data.

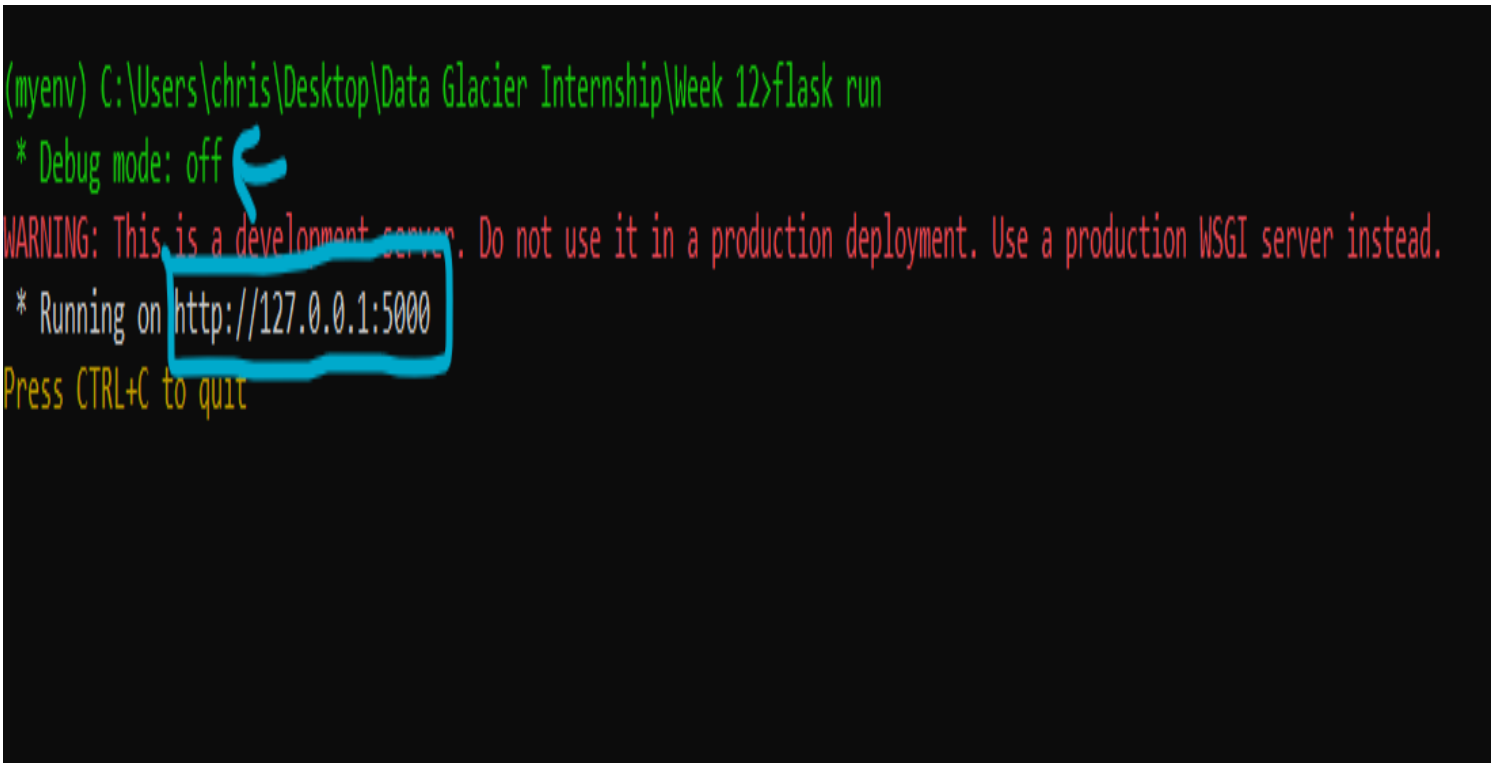
Model Deployment with Flask

Our next step it to create a Flask application to run our model on, and a web page to display it. The files `app.py' and `template/Index.html' do exactly that and are stored in the following link: Hate-Speech-Detection/Week 12 at main · cchristoforou97/Hate-Speech-Detection (github.com).

Now, we only need to run the application on the command line. However, keep in mind the following:

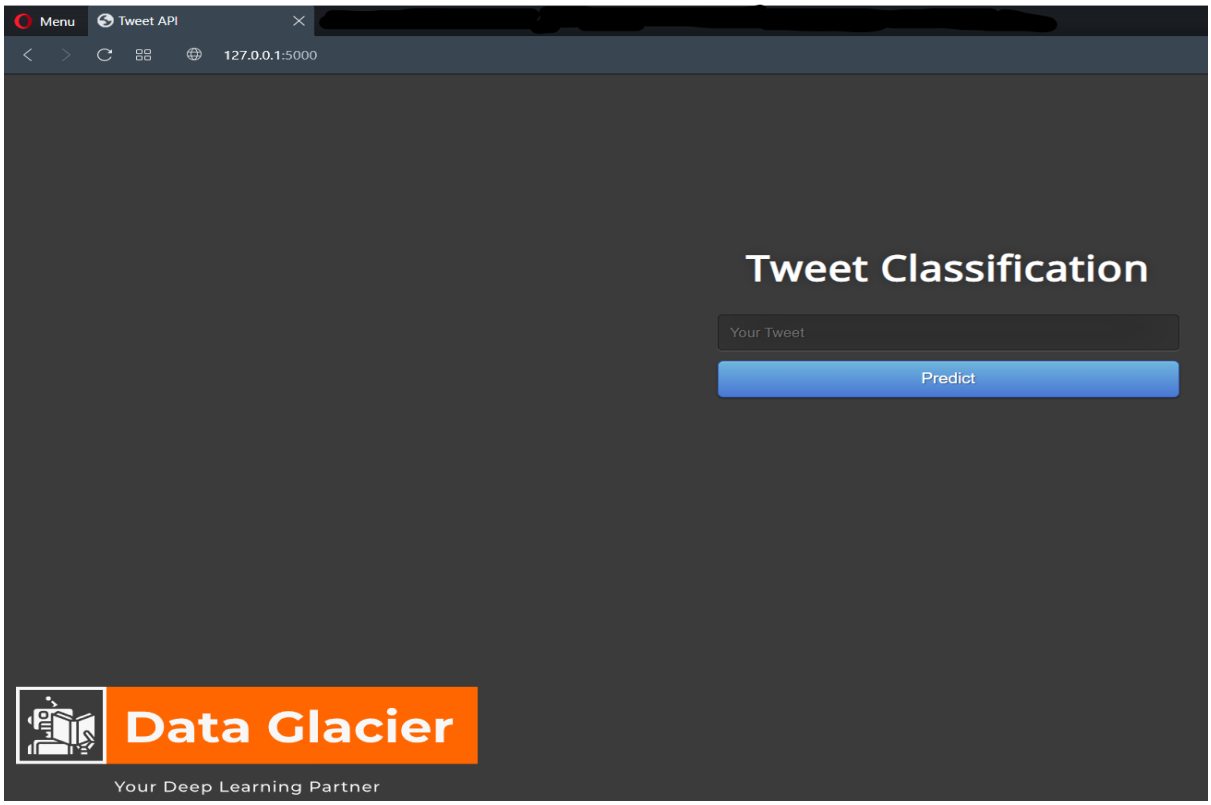
- The Torch library is used, which is compatible only with the python versions 3.7-3.9. Hence, be careful to use an appropriate version.
- You may encounter issues related to the 'multiprocessing' module while trying to run the application. There are a few reasons this may be happening (e.g., the python version, the simpletransformers library). To avoid these issues just turn off the debugger.

Now, we are ready to run the application on the command line. We will use the `flask run' command instead of the `python app.py' command to solve the debugger issue, we mentioned earlier.

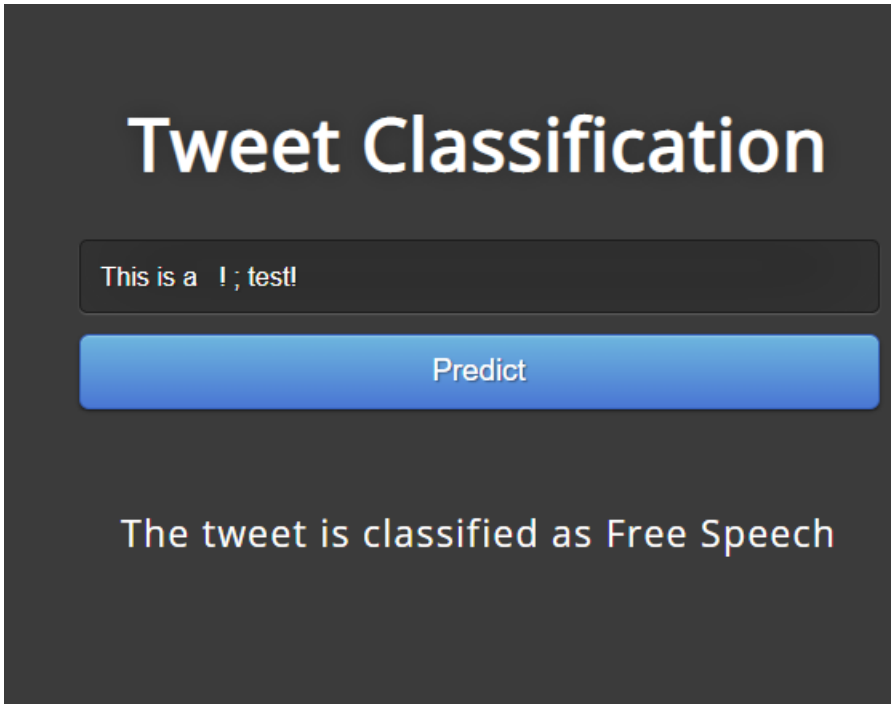


Next, we can open a web browser and navigate to our application's website using the provided URL: http://127.0.0.1:5000.

Here is our simple website, where you can enter your tweet and then press the `Predict' button to check if it is `Free Speech' or `Hate Speech'.



Let's see an example.


Tweet Classification
This is a ! ; test!
Predict
The tweet is classified as Free Speech