**Stranger thing tweets sentiment analysis**

**AML-2304 Natural Language Processing**

Submitted To

INSTRUCTOR: Parisa Naraei

CLASS: AML-2304

Ambika Sharma - C0848349  
Artificial Intelligence And Machine Learning   
Lambton CollegeToronto, Ontario, Canada

Priyanka Raskonda – C0851343   
Artificial Intelligence And Machine Learning  
Lambton CollegeToronto, Ontario, Canada

Allen Richy – C0837008  
Artificial Intelligence And Machine Learning  
Lambton CollegeToronto, Ontario, Canada

*Abstract*—*Twitter provides a wealth of data on people's preferences. Extracting information from data generated on social media has become a significant academic subject because of social media's increasing acceptance and appeal. These vast volumes of data are used to create models that forecast trends and behaviour. Nowadays, after any drama-series or TV show is released, people try to give their opinion through tweets. PyTorch Bi-LSTM RNN and Naive Bayes baseline models’ approach were put forth in this study to forecast sentiments of people towards the Stranger Things web series through tweets.*

*Keywords—Twitter, social media, PyTorch Bi-LSTM RNN, and Naive Bayes baseline model*

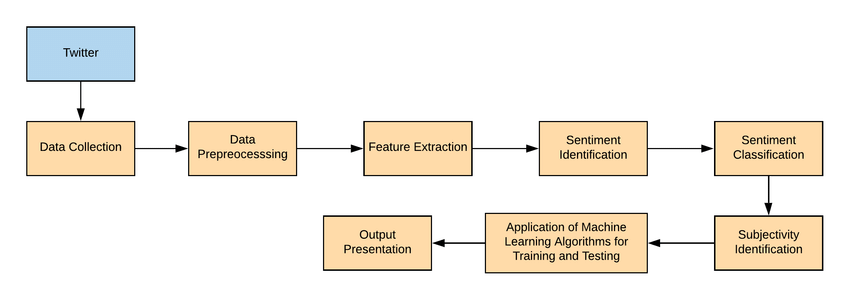
# **Introduction**

# **(*Motivation:*)**

A key area of machine learning called sentiment analysis seeks to extract subjective data from textual evaluations. The study of sentiment is strongly related to text mining and natural language processing. It can be used to assess the reviewer's perspective on specific subjects or the review's overall polarity. Using sentiment analysis, we may determine the reviewer's emotional state at the time of writing and determine if the review was favourable, negative, or neutral. In this project, we utilize sentiment analysis to examine a collection of drama-series evaluations written by reviewers to determine how they felt about the film overall—that is, whether they enjoyed it or despised it.

Given the wealth of information available on Twitter and the complexity of the current drama-series-rating systems, our study goal was to make them simpler. We intended to create a system that could determine whether the general public's view of a certain drama series is positive, negative, or somewhere in between using the 140 characters (or even less) that Twitter users write on their profiles. We divided opinions into three categories: positive, neutral, and negative. This also goes by the name of sentiment analysis. Our research was able to provide a good depiction of the public opinion about a drama series by extracting the viewpoint of a sizable number of people using hashtags, even though Twitter users are often average viewers who are not demanding.

The architect of the whole procedure done in the project is shown in the figure.



The deep learning model for prediction

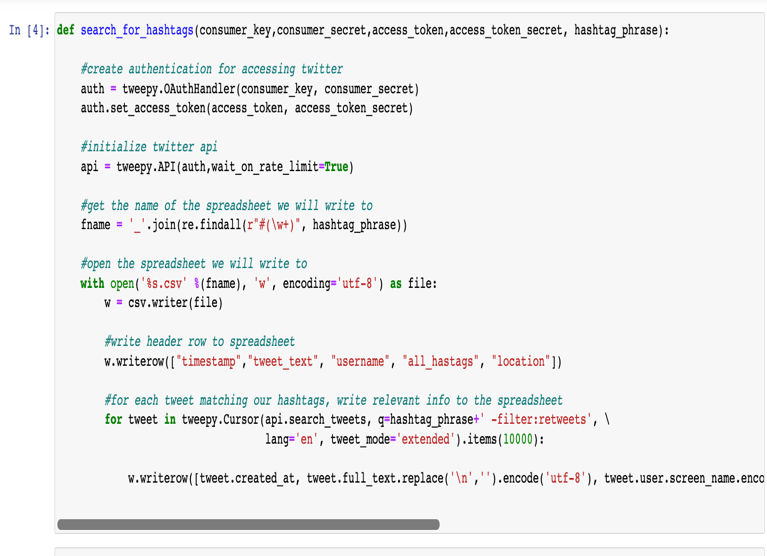
# **experiments:**

## Scrapping the dataset from Twitter

The first step that we had done is that we had scrapped the dataset from Twitter. For this, we extracted the tweets using the #StrangerThings. For this, we used the tweepy library of python. An open-sourced, user-friendly Python library for interacting with the Twitter API is called tweepy. It provides you with an interface so that your Python program may use the API. Hence, using this library we extracted timestamp, text tweet, hashtags, username, and follower\_count. After this, we saved the dataset as a .csv file. Hence, utilize our own created dataset for analyzing the sentiments of people regarding the Web series called Stranger Things. The description of the dataset is given below:

1. Timestamp – This column contains the date and time on which the tweet was created.
2. Tweet\_text – tweets created by people about stranger things.
3. Username – the name of the person who had tweeted.
4. All\_hashtag – all the hashtag words that are available in the text.
5. Follower\_count – number of followers of that person.

We had defined the function for extracting the things from tweeter as explained above in this function we used tweepy for getting the values from Twitter API. This is depicted in the underlying screenshot.



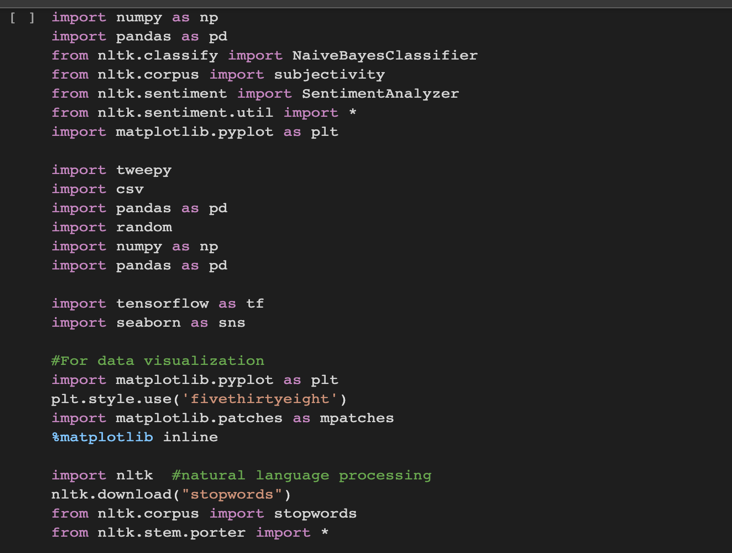
After this, we saved the extracted elements in csv which will act as our dataset for our further visualization and modelling.

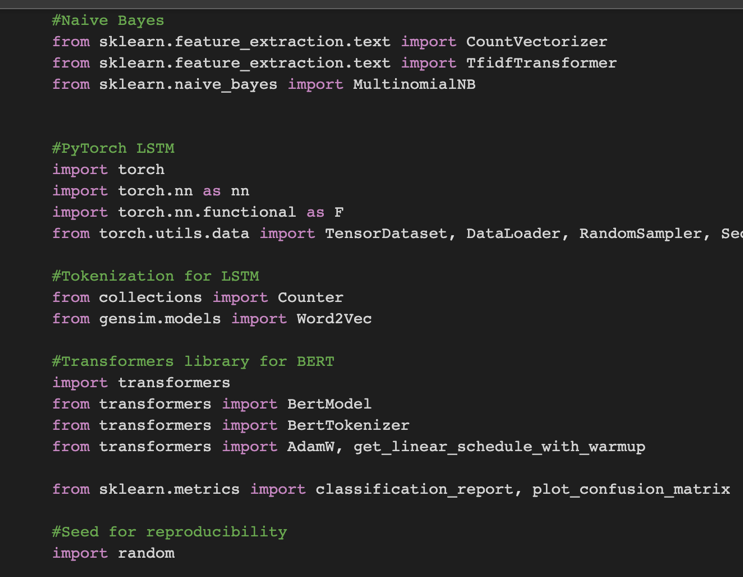
Graphical user interface, text, application

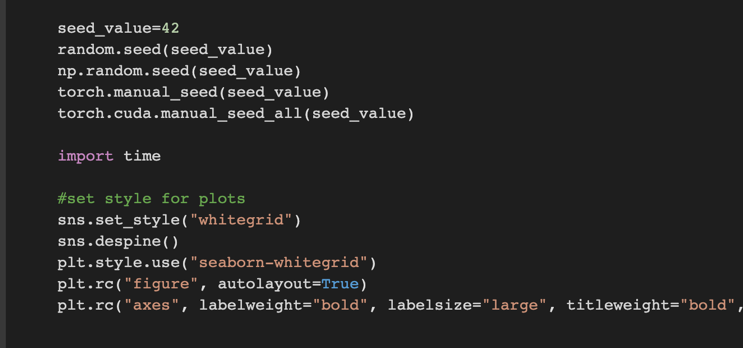
Description automatically generated

## Important libraries

After getting the desired dataset, we started working on the next step of the architecture which is visualizing the data, for this we had imported the import libraries like numpy , pandas, nltk, tweepy, random, tensorflow, matplotlib, stopwords, navies byes, Bert and Pytorch Lstm.







## Cleaning the data

Before diving our dataset into training and testing, we took a few steps for cleaning and managing the dataset like cleaning all the HTML tags from the text for this, we used the beautiful soup library of NLTK, cleaned the strings that were free from URL addresses, removed all the punctuations, removed whitespaces using .strip(), with the help of re.sub() module, removed all the accented characters like NKFD from text using normalize() function and last removed all the @mentions available in the text using regex.sub().

Text

Description automatically generated

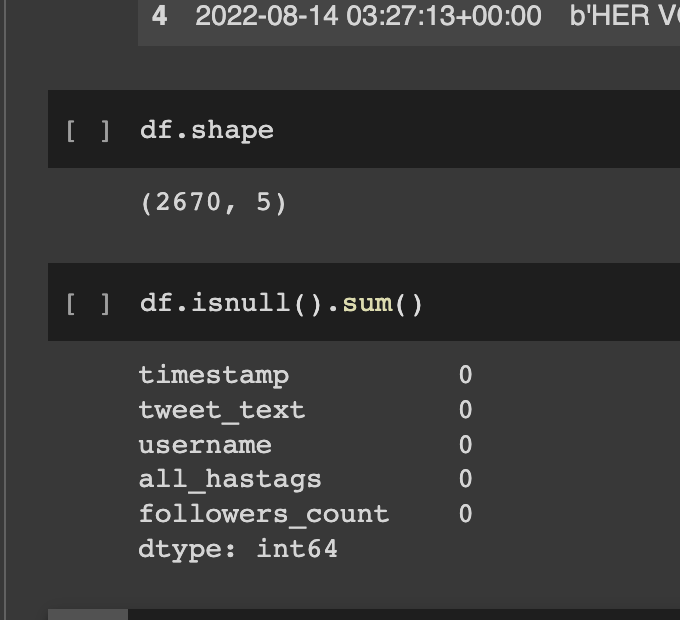
After cleaning our selected column looks like the below screenshot. Also after getting the cleaned column from all the HTML tags, punctuations, URL, @mentions and many more available unwanted characters that were of no use we set all the words in lower case for better readability and to get the best results for analyzing the sentiments of people.

Text

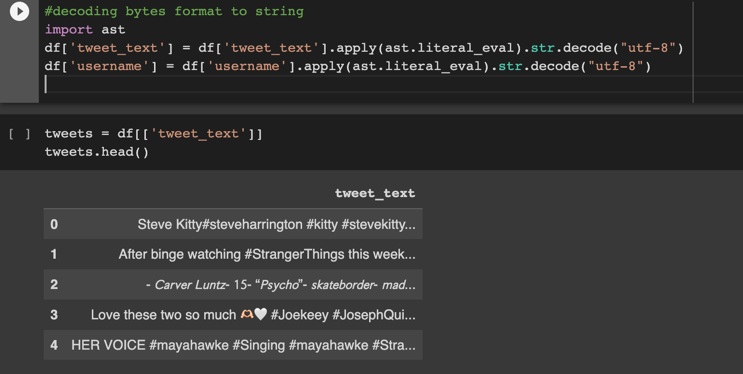
Description automatically generated

## Features

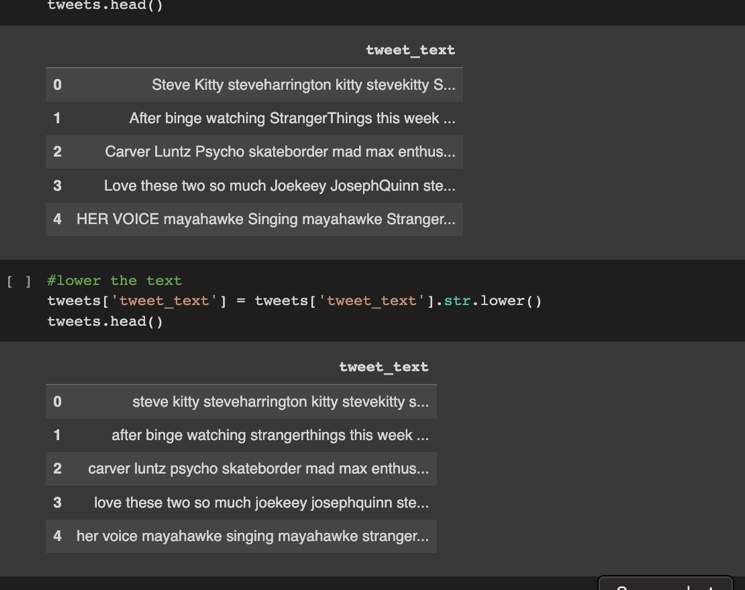
After importing the required libraries, we started visualizing our datasets like missing values and the shape of the dataset. We used to shape() for depicting the shape of the dataset and got to know that the dataset contains 2670 rows and 5 columns. Also, our dataset didn’t contain any missing values.



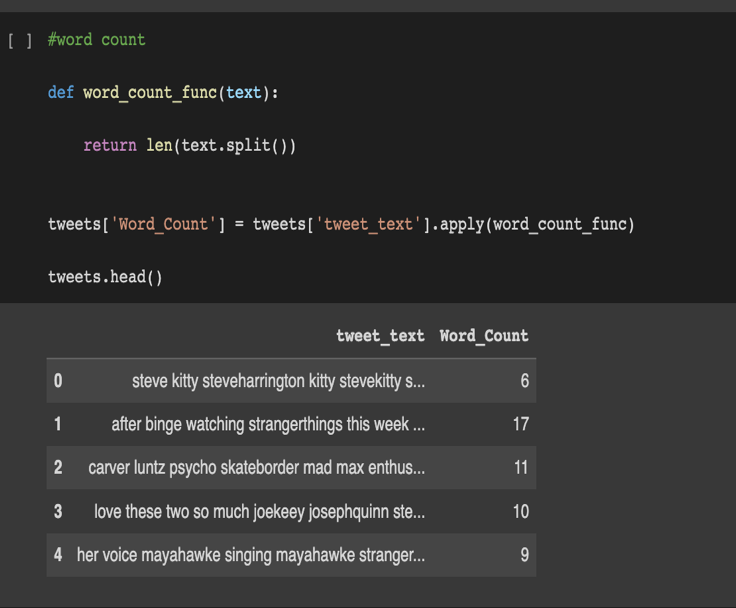
After depicting shape and missing values. We selected our targeted column that is tweet\_text as other columns is of no use for analyzing the sentiments of the people. So, we dropped all the columns. Before selecting our desired column, we decode the byte format of this column to string using ast module. Python programs can handle trees of the Python abstract syntax grammar with the aid of the ast module.

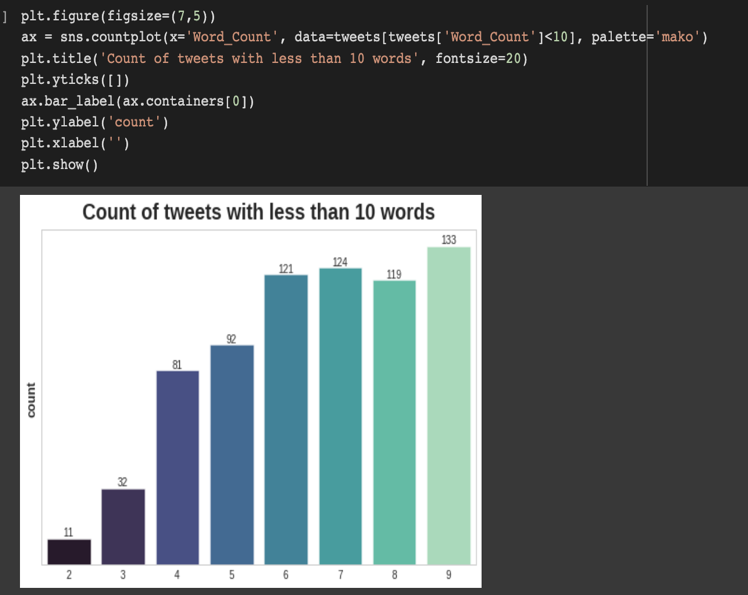


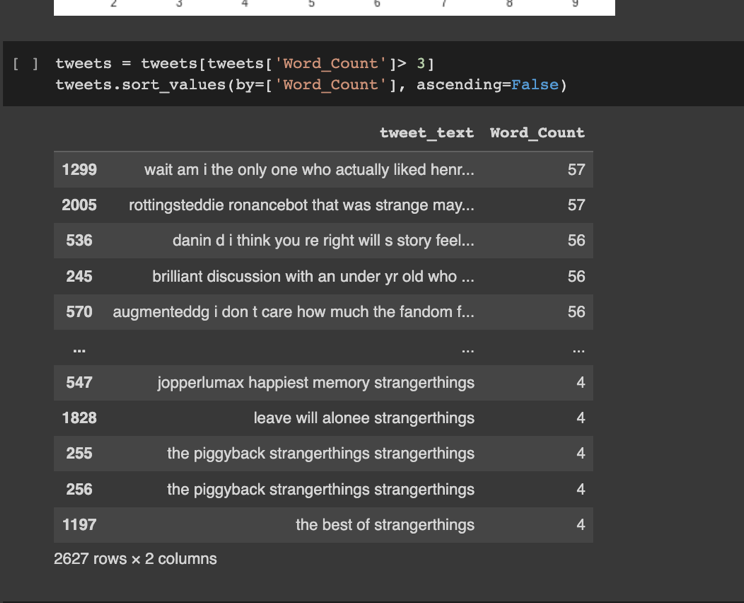
After cleaning our selected column looks like the below screenshot. Also, after getting the cleaned column from all the HTML tags, punctuations, URL, @mentions and many more available unwanted characters that were of no use we set all the words in lower case for better readability and to get the best results for analyzing the sentiments of people.



After this step, we went for the word count for each text and visualized the number of rows that has words less than 10. Also, we visualized that has more than 3 words.







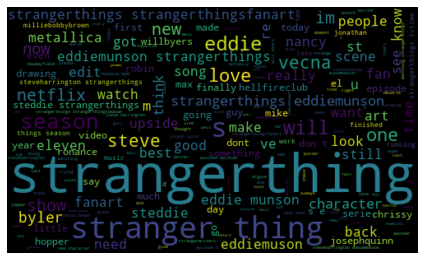
Chart

Description automatically generated

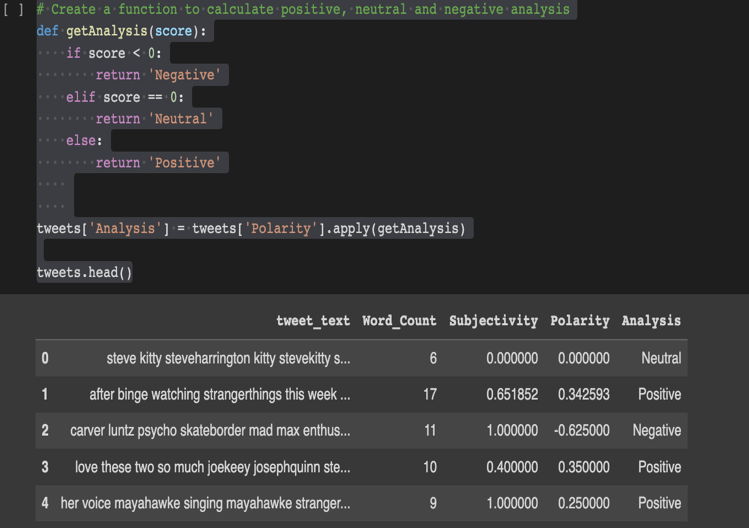
For creating the wordcloud, we had added two columns along with the text\_tag that contains the text, and word\_count that represents the number of words in the text, along with these two we added subjectivity and polarity. The term "polarity" describes how strong an opinion is whereas subjectivity refers to expressing some unique emotions, opinions, or convictions. After this, we tried to build the word\_cloud based on these two additional columns. The magnitude of each word in a word cloud, a data visualization technique for expressing text data, shows its frequency or relevance. Using a word cloud, significant textual data points can be highlighted. Word clouds are frequently employed for social network data analysis.

Text

Description automatically generated



After plotting the word cloud, we defined a function for analyzing whether the text is positive, negative or neutral. After this using this dataset we build a wordcloud of positive words, negative words and neutral words which look like underlying figures.

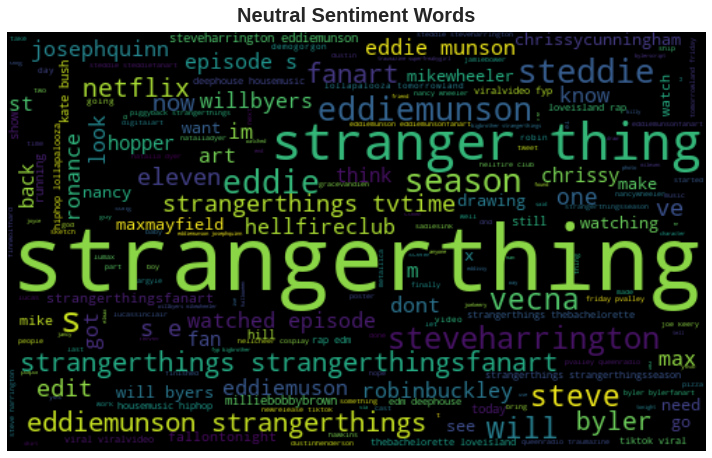


Text, chat or text message

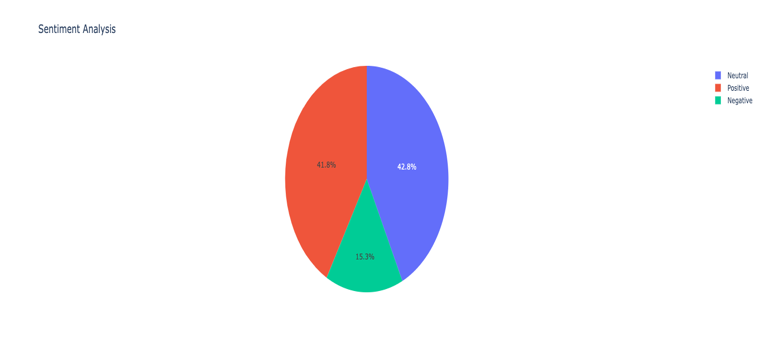
Description automatically generated

Text, chat or text message

Description automatically generated



After this, we visualized the percentage of each of the sentiments and found that 41.8% of texts belongs to positive sentiments, 42.8% of tweets were of a neutral tone, however, 15.1% of tones were negative.



We counted the number of words count that are regarded as positive, negative, and neutral. So, we analyzed that 1125 words are neutral, 1099 are positive words whereas it contains only 403 negative words.

#### 

## Splitting our dataset into train and test data

We split our dataset into train and test. In our dataset, as our target column is an analysis in which the sentiments of people are depicted so we took that as the target along with the predicted column that is Tweet\_text. After this, we look the shape of the targeted and predicted column which was equal to 2627.

A picture containing text

Description automatically generated

Graphical user interface, text

Description automatically generated

## Modelling

#### Every time we look into an ML model, we want to see if it outperforms the competition, especially if there is a more straightforward or manageable method. The baseline refers to the current strategy. As a starting point model, naive Bayes is frequently a wise choice.

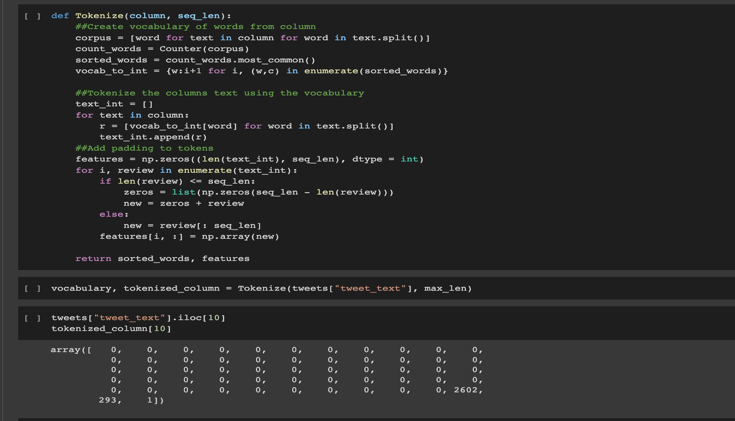
#### Every time we research a new task, we want to know if there is even a remote chance that an ML method can be successful. Before attempting a more complex model, it is frequently worthwhile to try a simple one first, such as Naive Bayes. While not an exact match, this is undoubtedly relevant to creating a baseline.

#### Graphical user interface, text Description automatically generated

The two LSTMs that make up a bi-LSTM are referred to as "forward LSTM" and "reverse LSTM." The forward LSTM basically receives the sequence in the initial order, whereas the backward LSTM basically receives the sequence in the reverse order.

We had defined the function tokenize that creates the vocabulary of words from the columns and tokenizing them and then, padding them to tokens and hence visualizing it in the form of arrays. In a manner similar to how we pre-processed the data for Naive Bayes, we must tokenize the phrases using a specially designed function.

To ensure that each statement contains the maximum number of words, the sentences will be transformed into lists of numbers.

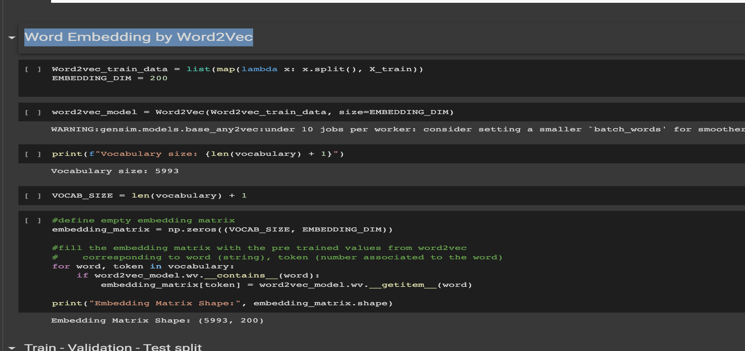


After creating and finding out the vocabulary in the words, we visualized the top 20 most common words.

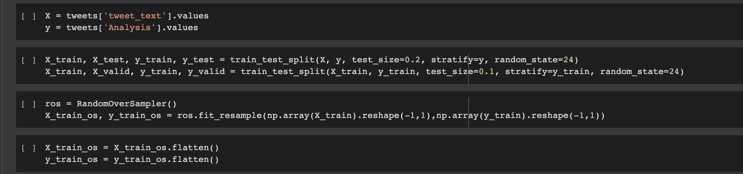
Chart

Description automatically generated

Word embedding by word2vec: A group of algorithms called Word2Vec can create word embeddings. Word embeddings are vectors that depict the spatial locations of words' semantic meanings. It is a deep learning model. The pre-trained model Word2vec and the original text tweets will be used to generate a word embedding matrix next. From the previously produced X train vector, we first generate a list of words.  We defined an embedding word dimension, which can be thought of as the number of features added to each converted word.  The training words and selected embedding dimensions are then passed to the imported Word2Vec object to initiate the Word2Vec model.  We must decide on the maximum number of words before defining the embedding matrix. From the previously built Python vocabulary dictionary, we will retrieve the total amount of terms. we calculated the vocabulary size which came to 5993 and then calculated the embedding matrix with the word and vocabulary that came up with (5993, 200).



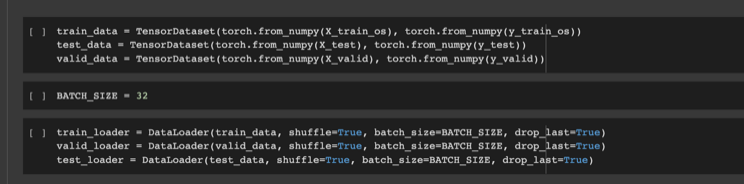
We will now generate training, validation, and test datasets using the tokenized texts.



Text

Description automatically generated

To enable batch data extraction for LSTM training, validation, and testing, the three sets will be converted to tensor datasets and data loaders.



After setting up some hyperparameters like num\_class as 5 as we were working for multiple classes, bidirectional setting to true, as we used bi-LSTM. We had now created a unique training loop, added an early stopping feature, and only stored the most accurate models for validation.

Text

Description automatically generated

Text

Description automatically generated

Text

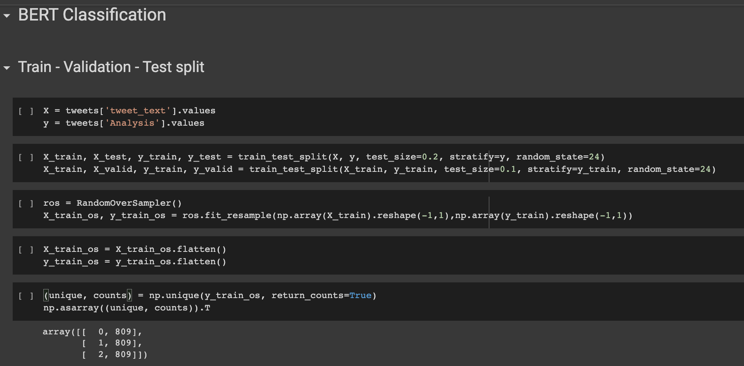
Description automatically generated

On the final cycle, our accuracy came up to 64%. We also calculated the report for a bi-LSTM model for each sentiment and calculated precision, recall, F1-score and support.

Text

Description automatically generated with low confidence

Then, after fine-tuning it for our classification objective, we loaded a pre-trained BERT model from the Hugging Face collection. First, because we wanted to tokenize the phrases differently than before, we divided the dataset into the train, validation, and a second test (Naive Bayes and LSTM). We loaded the specific BERT tokenizer from the Hugging Face library since we needed to tokenize the tweets for BERT. The loaded tokenizer was then used to define a special tokenizer function. We used the "encode" method of the original BERT tokenizer to tokenize the train tweets since we wanted to specify the length of the longest tokenized sentence before determining which phrase was the longest. The train, validation, and test tweets were then tokenized using the custom-defined tokenizer. Then, we defined the Cross-entropy Loss function for the multiclass classification task and hence calculated the accuracy.

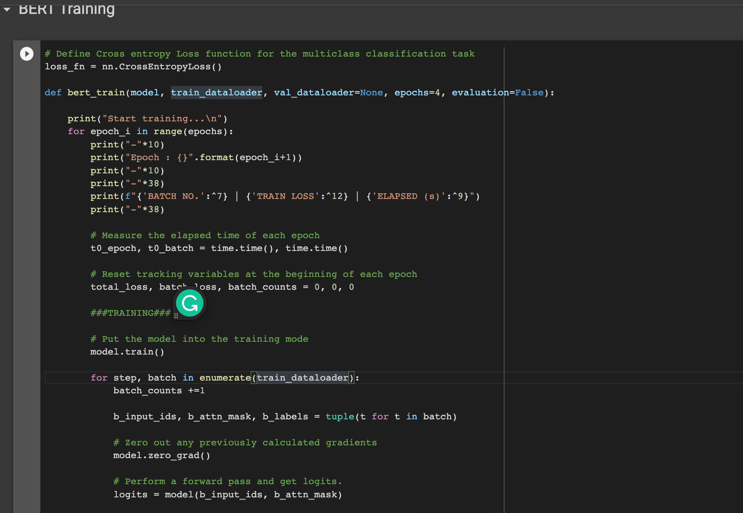


*Text

Description automatically generated*

Text

Description automatically generated



# Results

We created the StrangerThings.csv dataset by extracting the tweets of people from Twitter using tweepy, to analyze their sentiments toward people on a Tv Show named Stranger Things. After pre-processing the dataset, we used three models for getting the accuracy of predicting the sentiments of people over this web series. The results that we got from all these models are as follow:

1. Naïve Bayes – This model didn’t show much accuracy in analyzing the sentiments of people

Calendar

Description automatically generated

1. Bi-LSTM – This model showed greater accuracy than Naïve Bayes, above 70% at the 14th epoch.

Text

Description automatically generated

Calendar

Description automatically generated

1. BERT – This model outperformed in gaining accuracy. It achieved an accuracy of 84 % at epoch 25th. Hence, it was able to predict the sentiments of people towards the web series name Stranger Things.

Table

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with low confidence

# Discussion

After we delivered our proposal and got approval from the professor. We set a meeting on teams and discussed each other roles and gave them the time slot which is shown in the Table and hence were able to finish this project before time. Each of us has put equal effort into gaining the best results.

| Table 1 | | |
| --- | --- | --- |
| Names | Duties assigned | Timeslot |
| Ambika Sharma | Extracting Dataset + Report | 1 August to 2 August  15th August |
| Priyanka Raskonda | Modeling + Presentation | 8 August to 11 August  15th August |
| Allen Richy | Cleaning | 3 August to 7 August |

# Conclusion

In Conclusion, we found that people have neutral sentiments towards the web series Stranger Things, the least people have negative thoughts regarding this TV show whereas people who have positive thoughts begged the second position. Hence, this TV show was favoured by most people. Out of all the three models that we had performed which are Naïve Bayes, Bi-LSTM and BERT, BERT gained the maximum accuracy in predicting these sentiments of people towards the Stranger Things web series.

# References

AGARWAL, R. (2018). *BiLSTM - Pytorch and Keras*. Retrieved from Kaggle: https://www.kaggle.com/code/mlwhiz/bilstm-pytorch-and-keras

Horev, R. (2018, 11 10). *BERT Explained: State of the art language model for NLP*. Retrieved from towardsdatascience: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Yıldırım, S. (2020, 06 05). *A Practical Guide for Exploratory Data Analysis: Drama-seriess on Streaming Platforms*. Retrieved from towardsdatascience: https://towardsdatascience.com/a-practical-guide-for-exploratory-data-analysis-drama-seriess-on-streaming-platforms-5ea494fee9d2