Problem Description

Twitter is one of the most widely used social media channels today and is unique in hosting everyone from top politicians and movie stars posting influencer content to regular people chatting and sharing views on trending topics. The far-reaching effect of each tweet makes their study of great importance to understand the status quo.

Sentiment of a tweet is central to the message conveyed by the tweet. Studying the patterns of positive and negative sentiments in tweets can also provide insights into the popular understanding of common issues. In the project as well, we shall try to predict whether a tweet is of a positive or negative type, which is represented by 0 and 4 respectively.

Data Description and Preprocessing

The dataset used for the task of sentiment classification was divided into the following columns:-

- **Class:** The class represents the label assigned to each tweet. Each tweet was assigned a value of 0 if it were negative, and 4 if it were positive in sentiment.
- **ID:** This stood for the tweet id of each tweet. This value is unique to each tweet and is paramount in removing duplicates.
- **Date:** This column contains the date and time at which the tweet was tweeted in the form of a string.
- **Flag:** This column represented the status of each tweet, regarding whether there was any problem with the tweet. However, this was not of use because all the tweets were free from any problem in the dataset and had the value of 'NO_QUERY' throughout.
- **User:** This column contained the account id of the author of each tweet in the form of a string.
- **Tweet:** This column contained the tweet itself in the form of a string.

	Class	Id	Date	Flag	User	Tweet
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

Figure 1: Raw Data with Column Names

The data requires some preprocessing to make it fit for modeling. The steps for elementary data cleaning are as follows:

1. **Name the columns:** This step involves naming the columns with the names explained above.
2. **Convert to DateTime:** The dates present in string format were converted into python DateTime format to enable easy utilization of the temporal data in python.
3. **Drop surplus columns:** The surplus column 'Flag' was dropped since all values were the same, and didn't help in sentiment classification.
4. **Remove duplicate rows:** Duplicate tweets were removed by leveraging the unique tweet id so that we are left with only a single instance of each tweet.
5. **Convert tweets into lower case:** All tweets were converted into the lower case since they didn't affect the meaning of words or the sentiment of the tweet much. This would also allow

better clubbing and calculating the frequency of different words in the dataset, regardless of semantic differences.

6. **Remove punctuations:** All the punctuations were removed to get the raw tweet. However, special care was taken to retain special characters and emojis in the tweet, since they are significant for sentiment determination.
7. **Remove letters occurring more than 3 times consecutively:** All the letters which repeated more than 3 times in any word consecutively were replaced by only 1 instance of the letter since 3 consecutive letters are very uncommon in English.
8. **Remove @, hashtags and URLs:** The words containing @ and # and URLs were removed from the tweets to enable better retention of important words.
9. **Remove numbers:** All the information in the form of numbers was removed from the tweet, as it didn't provide qualitative insight into the tweet.
10. **Remove stop words:** All the semantic words with no apparent impact, like articles and prepositions, were removed from the tweet to get smaller tweets and support easier computations. However, special care was taken to retain words like 'not' since it is highly representative of tweet sentiment.
11. **Lemmatization and Stemming:** The words were simplified into their simplest contextual form (lemmatization) and root form (stemming). This allowed easy correlation between different forms of the word and facilitated the formation of good correlations between tweets.

Tweet	TweetMinusStopWords
is upset that he can't update his facebook by ...	upset not update facebook texting it might cry...
@kenichan i dived many times for the ball. man...	dived many times ball managed save rest go bo...
my whole body feels itchy and like its on fire	whole body feels itchy like fire
@nationwideclass no, it's not behaving at all....	no not behaving all mad here not see there
@kwesidei not the whole crew	not whole crew
...	...
just woke up. having no school is the best fee...	woke up school best feeling ever
thewdb.com - very cool to hear old walt interv...	cool hear old walt interviews ♫
are you ready for your mojo makeover? ask me f...	ready mojo makeover ask details
happy 38th birthday to my boo of all time!!! t...	happy th birthday boo time tupac amaru shakur
happy #charitytuesday @thenspcc @sparkscharity...	happy

Figure 2: Final Tweets after Cleaning

Data understanding:

1. Word Cloud: Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. To analyze the data, we plotted word cloud for the entire data set and for the 2 classes i.e. class 0 and class 4.

Figure 3.

For the entire data set

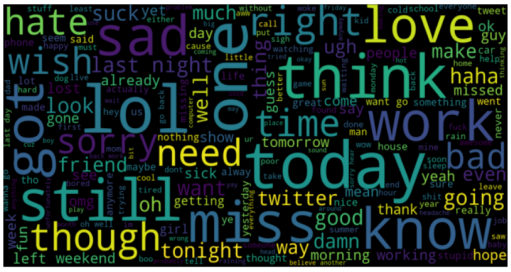


Figure 5.

Class 0 **Class 4**

Conclusion: We can say that class 0 is a negative class as words like sorry, hate, sad, need etc. are highlighted in the word cloud of class 0 and class 4 is positive class as words like good, thank, awesome, well lol are highlighted in the word cloud of class 4. To confirm this, we have plotted and analyzed n-grams for both classes.

2. N-grams: An *n*-gram is a contiguous sequence of *n* items from a given sample of text or speech. If n = 2, then it is bigrams and for n = 2, it is called trigrams.

Bigrams

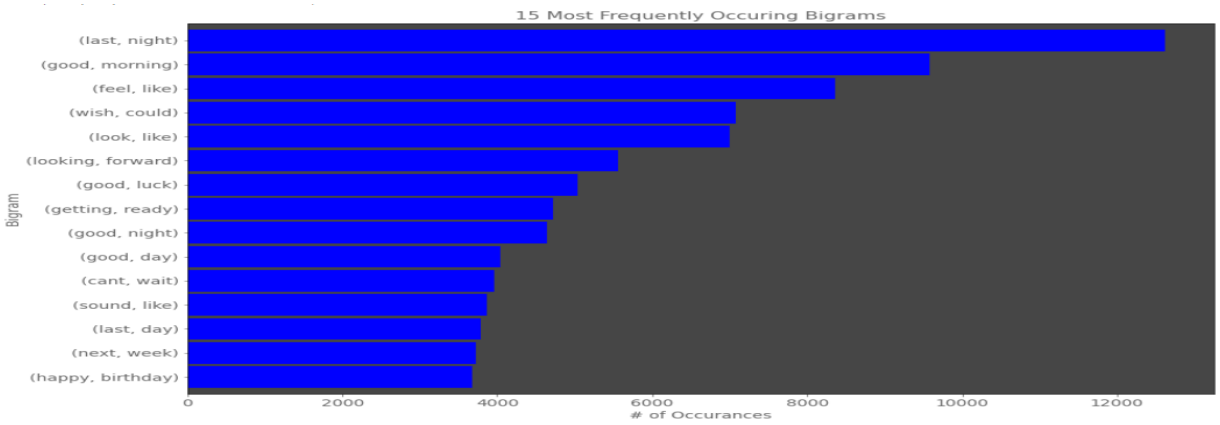


Figure 6.

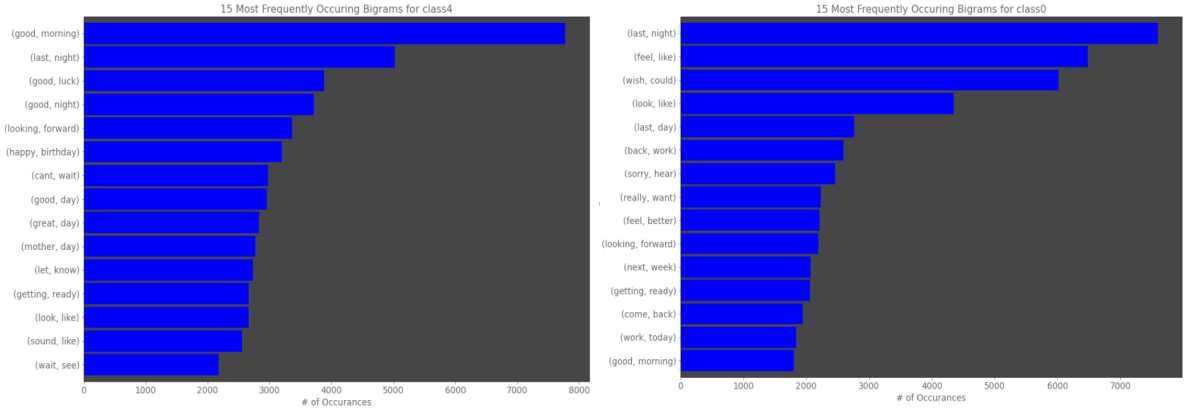
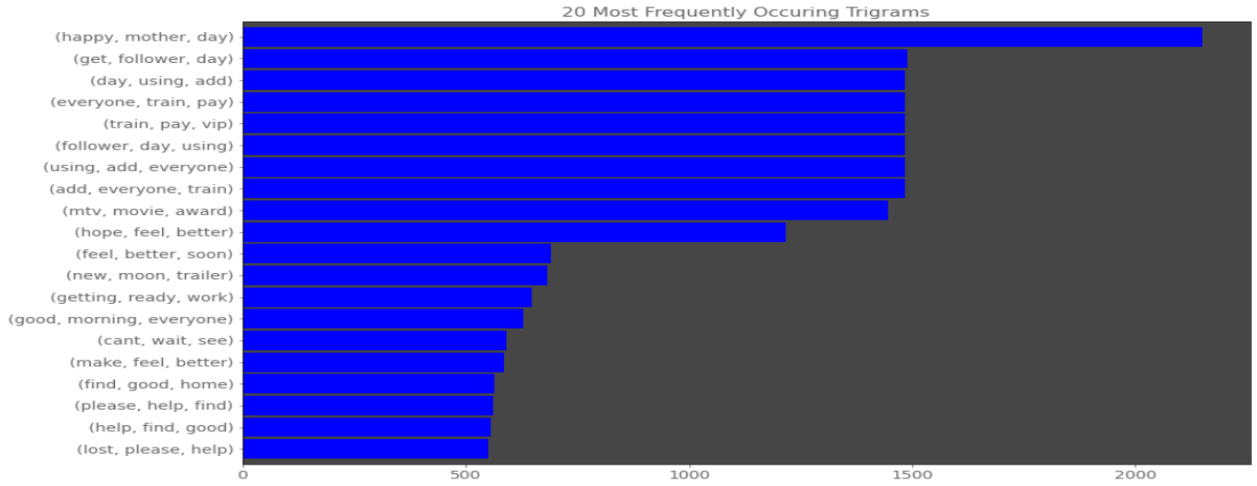


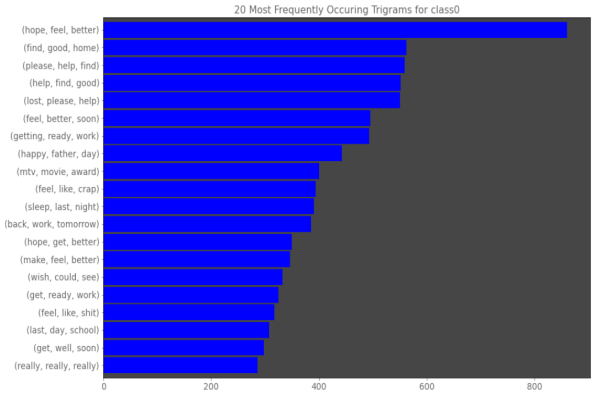
Figure 8.

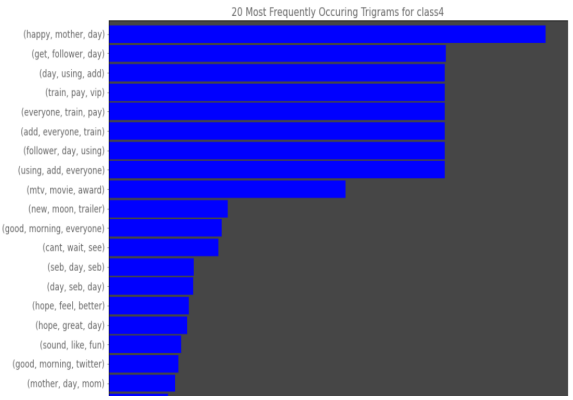
Class 4 **Class 0**

Trigrams



For the entire data set





Class 0 **Class 4**

From the n-grams, it can be concluded that people post their everyday life on Twitter. There were some interesting events that happened in 2009 (when this data was collected) which can be clearly seen in the analysis.

1. The trailer of the very popular movie Twilight: The new moon was released during that time and people have shown a very positive response to the movie.
2. This data spans over approximately 3 months that is from May to July, and during this time Mother's Day and Father's day are celebrated and that is quite evident from the data.
3. Happy birthday is one of the most tweeted tweets. It seems reasonable as every day is someone's birthday.

And above all, the distinction between the two classes becomes very clear.

Model Development

In our project, we have used four different models to train our dataset and get insights. The first have used TF-IDF word embedding techniques, while the last one has used the GLove word embedding method. These models are :

- Bernoulli Naive Bayes
- Linear SVM
- Logistic Regression
- Long Short Term Memory (LSTM) Neural network

Simple ANN models won't work on textual data. There are several reasons for it. First, inputs and outputs can be of different lengths. And then a second and more serious problem is that a naive neural architecture does not share features learned across different positions of text. In short, a simple neural network cannot differentiate between the same word with different meanings. To solve this, we can use RNN(Recurrent Neural Network). But RNN faces a vanishing gradient problem. So we end up using Bi-Directional LSTM.

To prevent the overfitting of our LSTM model, we added two dropout layers in different parts of the architecture. We also reduce the learning rate after every epoch based on the accuracy of the validation data using LRScheduler.

We used simple as well as complex models and tried to find the performances of each to get the best results. For model architecture, we used

1) Embedding Layer - Generates Embedding Vector for each input sequence.

2) Conv1D Layer - It is used to convolve data into smaller feature vectors.

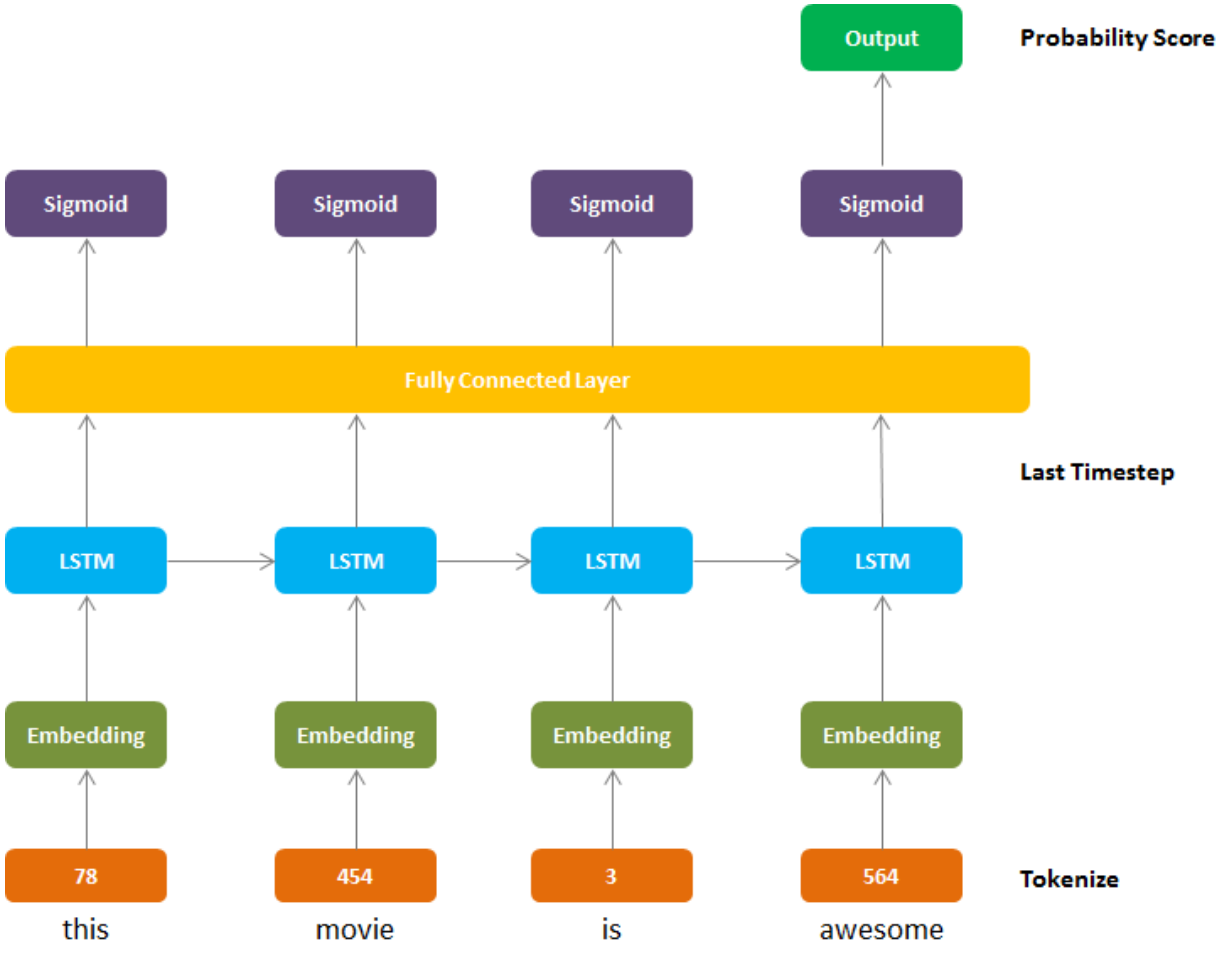
3) LSTM - Long Short Term Memory, is a variant of RNN which has a memory state cell to learn the context of words that are further along the text to carry contextual meaning rather than just neighboring words as in the case of RNN.

4) Dense - Fully connected layers of classification.

For optimization of our model, we used Adam optimizer.

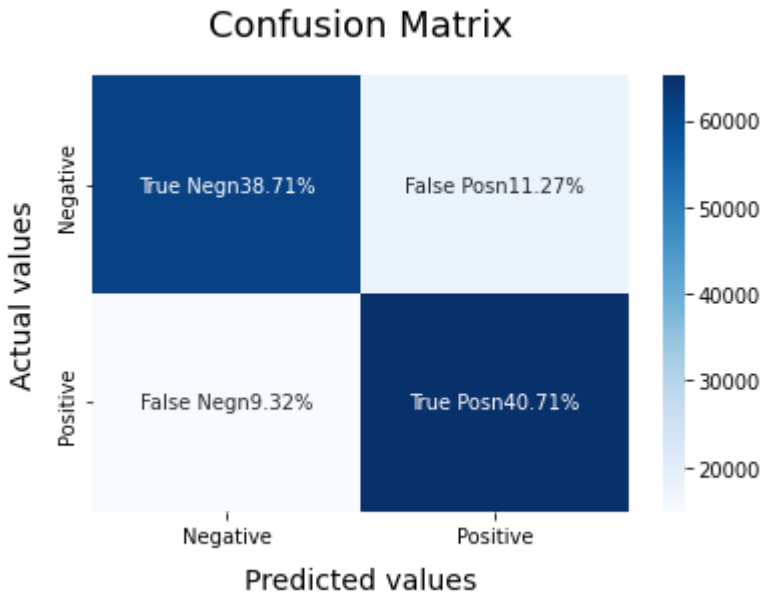
- Callbacks are special functions that are called at the end of an epoch. We can use any functions to perform specific operations after each epoch. We used two callbacks here.
- LRScheduler - It changes a Learning Rate at a specific epoch to achieve a more improved result. In this notebook, the learning rate exponentially decreases after remaining the same for the first 10 epochs.
- ModelCheckPoint - It saves the best model while training based on some metrics. Here, it saves the model with minimum Validity Loss.

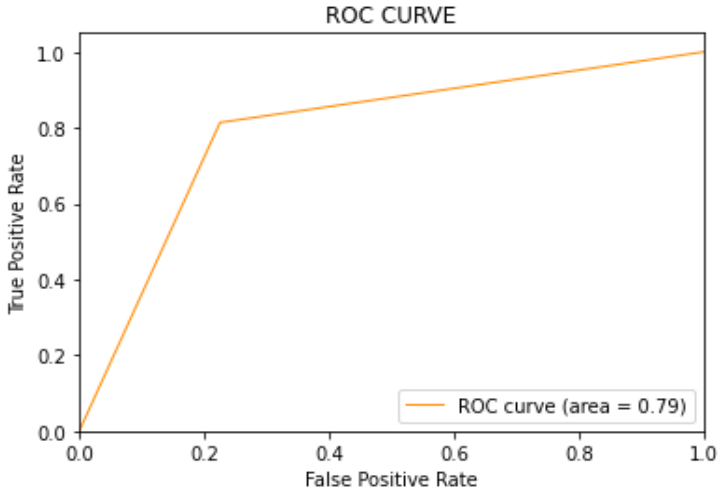
Model Architecture



Results

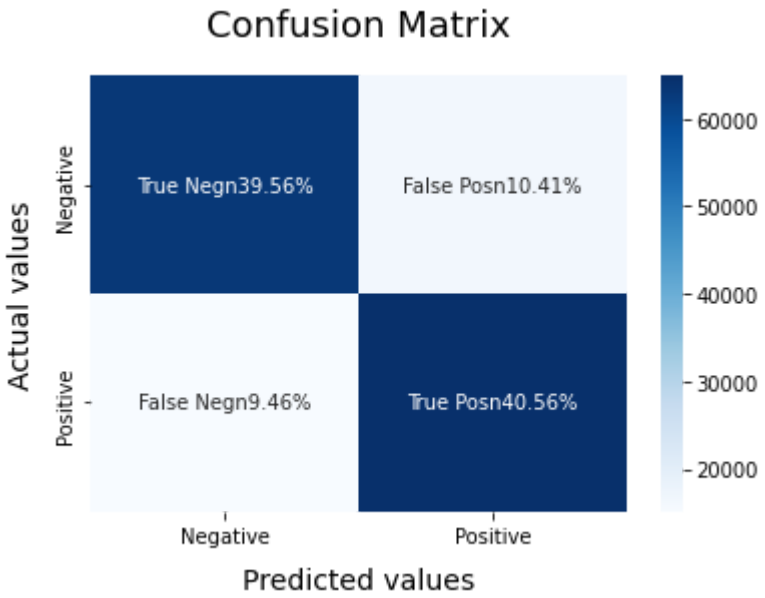
Bernoulli Naive Bayes:

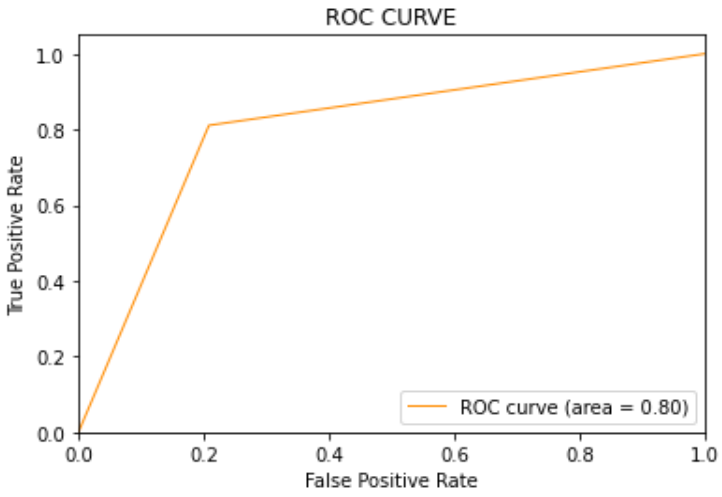




	precision	recall	f1-score	support
0	0.81	0.77	0.79	79962
1	0.78	0.81	0.80	80038
accuracy			0.79	160000
macro avg	0.79	0.79	0.79	160000
weighted avg	0.79	0.79	0.79	160000

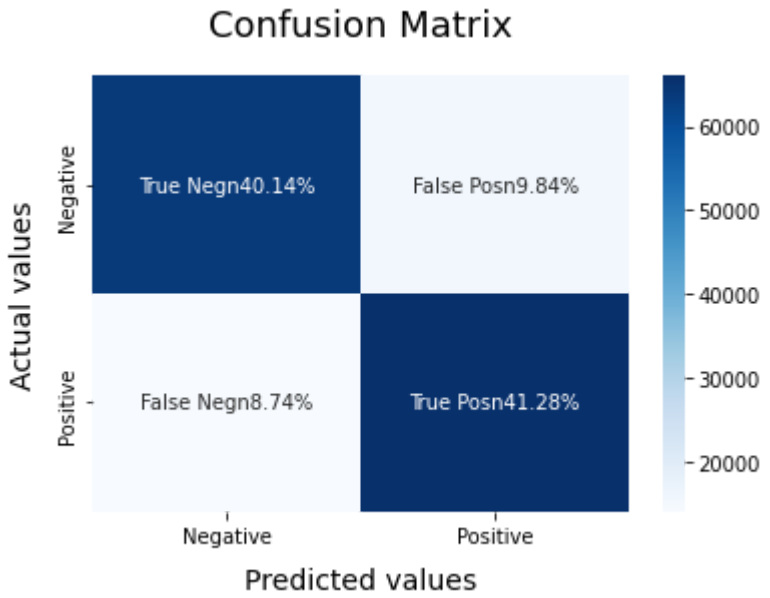
Linear SVM:

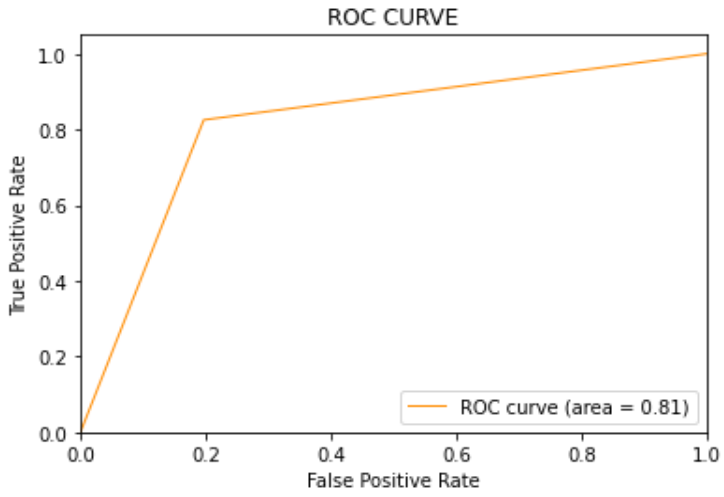




	precision	recall	f1-score	support
0	0.81	0.79	0.80	79962
1	0.80	0.81	0.80	80038
accuracy			0.80	160000
macro avg	0.80	0.80	0.80	160000
weighted avg	0.80	0.80	0.80	160000

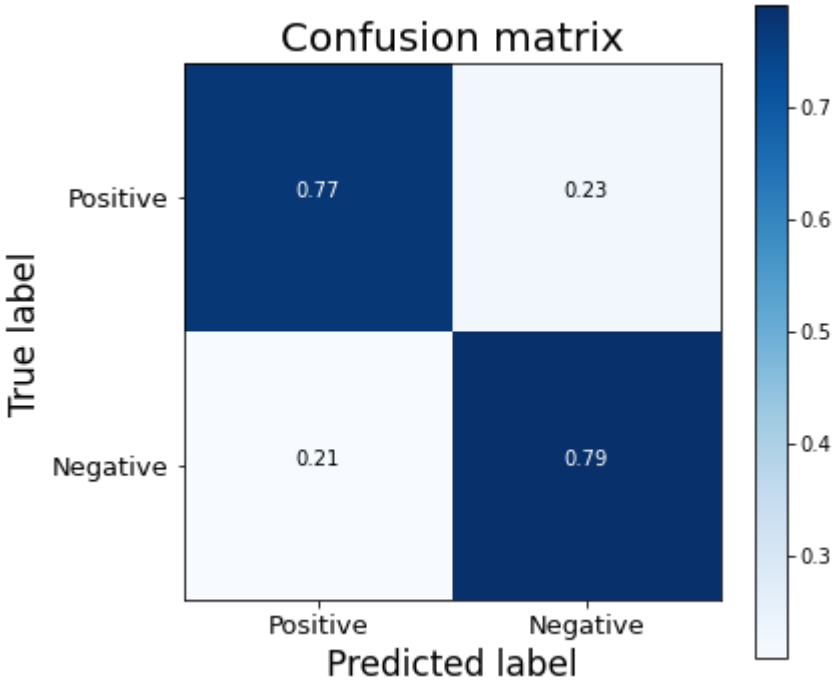
Logistic Regression:

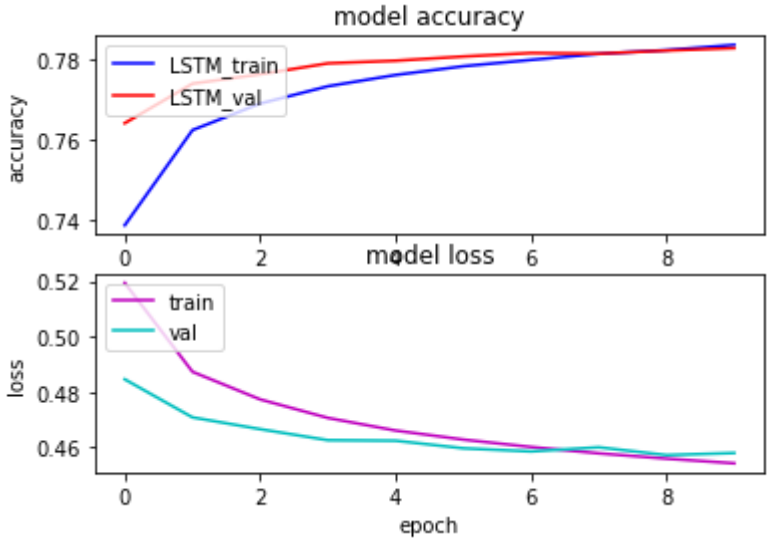




	precision	recall	f1-score	support
0	0.82	0.80	0.81	79962
1	0.81	0.83	0.82	80038
accuracy			0.81	160000
macro avg	0.81	0.81	0.81	160000
weighted avg	0.81	0.81	0.81	160000

LSTM(GLove):


Confusion matrix
Positive
Negative
True label
0.77
0.23
0.21
0.79
Positive
Negative
Predicted label
0.7
0.6
0.5
0.4
0.3


model accuracy
LSTM_train
LSTM_val
accuracy
0.78
0.76
0.74
0
2
4
6
8
model loss
train
val
loss
0.52
0.50
0.48
0.46
epoch

Conclusions

Embedding	Model	Precision	Recall	F1 score	Accuracy
Glove	LSTM	0.78	0.79	0.78	0.78
TF-IDF	Bernoulli Naive Bayes	0.78	0.81	0.80	0.79
	Linear SVM	0.81	0.79	0.80	0.80
	Logistic Regression	0.81	0.83	0.82	0.81

Results Interpretation

- [ ] The performance of all the models was similar
- [ ] Logistic regression and Linear SVM are slightly better models than other
- [ ] LSTM model can be improved by fine-tuning hyperparameters. Instead of LSTM, we can use Attention models can also be used.

Drawbacks and Future Work

- [ ] Our inability to do spell check due to limitations of Computational power
- [ ] Class different labels between training and test set leading to loss of information
- [ ] Lack of analysis of joint words like hashtags and slang like wanna