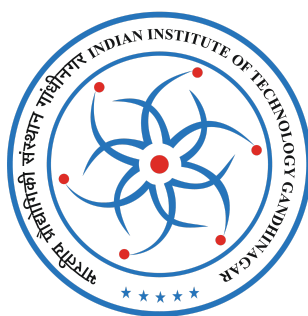


CS 328: Data Science WRITING ASSIGNMENT



Yash Kokane (Roll No.: 20110237)
Tanvi Chaudhari (Roll No.: 20110043)
Divya Chinchole (Roll No.: 20110060)

30th April 2023

-
- 1** Index
 - 2** Objective
 - 3** Part 1: Hate Speech Classifier
 - 4** Future Plans (for Part 1)
 - 5** Part 2: Analysing PM Modi's Tweets
 - 6** Link for notebooks and datasets

7 Objective

As part of the project, two independent projects have been carried out:

- Developing a hate speech classifier for Indian Social Media, in particular twitter, from scratch. Compare the models developed using Classical ML techniques and using a deep learning approach.
- Scraping tweets by Prime Minister Narendra Modi, and conducting an in-depth analysis on them.


The project was divided into two parts, as initially an attempt was made to develop a sentiment analysis model that can identify different types of hate speech, and then to use it on the scraped tweets. However, since the tweets by Prime Minister Modi cannot be classified to a positive or negative sentiment without a proper training dataset, it was decided to conduct the two projects independently.

For the sentiment analysis of Modi's Tweets, we have used the VADER (Valence Aware Dictionary and sEntiment Reasoner) library, since it can understand the contexts of the tweets much better than conventional sentiment analysis libraries like Huggingface or TextBlob. This is because it is specialized in analyzing sentiments in informal, social media posts.

A detailed explanation of the ML models and LSTM, in addition to the graphs plotted can be seen in the notebooks, with their links attached here:

Github Link for notebooks and datasets:

<https://github.com/yashkokane291/CS328-Datasets-and-notebooks.git>



8 Part 1: Hate Speech Classifier

Overview

As part of the project, a hate speech classifier has been developed, that can classify hate speech on the basis of Religion, Age, Gender, Ethnicity, and not hate speech. A dataset for the same, compiled by, as available on Kaggle was used, containing the aforementioned labels on tweets sourced from a wide variety of sources. Two approaches were taken to develop the hate speech classifier - the first one was a classical ML-based approach, where algorithms like Random Forest, SVM, Naive Bayes, Logistic Regression and K nearest neighbors were used. Preprocessing techniques were utilized before the algorithms were used, namely Cleaning, Stemming, and Lemmatization, Removing Stopwords, SMOTE and TF-IDF. Maximum accuracy was reached using Random Forests (94%). Further optimization did not lead to much improvement in accuracy.

For the deep learning approach, an LSTM (Long Short-Term Memory) neural network was used. The input to the model was preprocessed using techniques such as tokenization, removing stop words, and padding. Word embeddings were used to represent the words in the input data. The LSTM was trained on the preprocessed data using the Adam optimizer and categorical cross-entropy loss. The model was trained for 10 epochs, and a batch size of 64 was used. The best model was selected based on the validation accuracy, and the final accuracy achieved was 98%. Plots of accuracy vs epochs were made to visualize the performance of the model during training.

Additionally, word clouds were also made, showing the most common words associated with hate speech on religion, age, gender and ethnicity.

8.1 Machine Learning Approach

In this case, the hate speech classifier was trained using Naive Bayes, K Nearest Neighbours, SVM, Random Forest and Logistic Regression algorithms. The classification accuracy was determined in each case, by calculating the precision, recall and F1 score for each model. Additionally, confusion matrix were also plotted.

Pre-processing steps involved:

Different pre-processing techniques were used for the two approaches. This is because deep learning models can often benefit from more complex preprocessing techniques. Additionally, some deep learning models (such as convolutional neural networks) may be more robust to noise in the data and may require less aggressive preprocessing than some ML models. The deep learning models are better at understanding context, and might lose out on accuracy if the preprocessing utilized for ML approaches are used for them.

For the ML-based approach, the following pre-processing techniques were used.

- Text Cleaning: Here, any data that can act as noise for the dataset was removed, to reduce the tweets to simply individual words. Non-ASCII characters, newline and carriage returns, uppercase letters, punctuation marks, URLs, and hashtags were formatted into

plain lowercase text. Additionally, any empty rows in the dataset were also dropped.

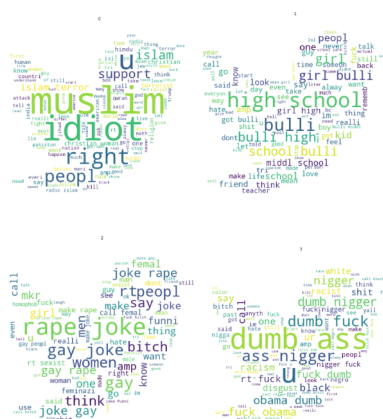
- **Lemmaization:** Lemmaization is the process of reducing a word to its base or dictionary form, known as the lemma. For example, the word "running" would be lemmatized to "run". This is useful in reducing the overall vocabulary size and ensuring that variations of the same word are treated as one.
- **Stemming:** It is the process of reducing words to their root or stem form. For example, "running" and "runner" would be stemmed to "run". This is useful in reducing the vocabulary size even further and in capturing the core meaning of words.

Both lemmatization and Stemming are used to simplify the text and make it easier for the machine-learning model to understand and analyze. They can help to reduce the complexity of the input text and can lead to better performance on NLP tasks.

- **Removal of stop words:** In the text cleaning process, stop words were removed using the NLTK stop words list. Stop words are common words in a language that do not carry much meaning and can be safely removed to reduce the dimensionality of the text data. This is done because stop words can add noise to the data and make it harder for the model to extract useful information.
- **SMOTE - Synthetic Minority Oversampling Technique:** The number of hate speech samples on ethnicity was more than on age, leading to an imbalance in the data. To avoid biased predictions, SMOTE was applied to the data, generating synthetic samples for the minority class. This technique helped to balance out the dataset and improve the performance of the classifier.

Word Clouds

Word clouds were developed for the preprocessed tweets, using the wordcloud library. The wordclouds for Gender, religion, ethnicity and age have been plotted below.



TF-IDF

TF-IDF stands for "Term Frequency-Inverse Document Frequency". It is a numerical statistic used to reflect how important a word is to a document in a collection or corpus of documents.

TF-IDF is a commonly used technique in NLP to represent text data in a numerical format that can be used in machine learning algorithms. The reason why it is done is that most machine learning algorithms require numerical input data to work, but text data is typically in a human-readable format of words and sentences.

TF-IDF helps to convert the text data into a numerical representation by assigning a weight to each word in a document. This weight reflects how important the word is to the document, by measuring how frequently it appears in the document (term frequency) and how rare it is in the corpus of documents (inverse document frequency). By using TF-IDF, we can convert each document into a vector of numbers that can be used as input to a machine-learning algorithm.

For each word, the weight can be calculated using the following formula:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$$\begin{aligned} tf_{ij} &= \text{number of occurrences of } i \text{ in } j \\ df_i &= \text{number of documents containing } i \\ N &= \text{total number of documents} \end{aligned}$$

Implementation

After preprocessing and TF-IDF, the data was ready to be fed to the ML models.

Here are the models (with a brief explanation of how they work) and their accuracies for classification. After testing, it was determined that Random Forest had the highest accuracy of 94%).

- Naive Bayes: Naive Bayes is a classification algorithm that uses probability theory to predict the likelihood of a given sample belonging to a particular class. It calculates the probability of each class based on the observed features and selects the class with the highest probability as the predicted class for a given instance. It assumes that the features are independent of each other, which simplifies the calculations.

	precision	recall	f1-score	support
religion	0.92	0.46	0.61	1587
age	0.9	0.75	0.82	1574
gender	0.76	0.8	0.78	1490
ethnicity	0.89	0.89	0.89	1547
not_hatespeech	0.45	0.77	0.57	1392
accuracy			0.73	7590
macro avg	0.78	0.73	0.73	7590
weighted avg	0.79	0.73	0.74	7590

- Random Forest: Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their outputs to make a final prediction. Each tree is built on a random subset of the data and a random subset of the features. The final prediction is based on the mode of the predictions made by the individual trees. Random Forest is robust to noise and overfitting, and can handle both categorical and continuous data.

	precision	recall	f1-score	support
religion	0.96	0.97	0.96	1587
age	0.98	0.98	0.98	1574
gender	0.95	0.86	0.9	1490
ethnicity	0.99	0.98	0.99	1547
not_hatespeech	0.81	0.89	0.85	1392
accuracy			0.94	7590
macro avg	0.94	0.94	0.94	7590
weighted avg	0.94	0.94	0.94	7590

- SVM: Support Vector Machines (SVMs) are a class of machine learning algorithms that can be used for classification, regression, and outlier detection. SVMs work by finding the hyperplane that maximizes the margin between the two classes in the feature space. SVMs can handle both linear and non-linear data, and can be extended to handle multiple classes and high-dimensional data.

	precision	recall	f1-score	support
religion	0.96	0.96	0.96	1587
age	0.95	0.97	0.96	1574
gender	0.95	0.86	0.9	1490
ethnicity	0.98	0.97	0.98	1547
not_hatespeech	0.79	0.86	0.83	1392
accuracy			0.93	7590
macro avg	0.93	0.92	0.92	7590
weighted avg	0.93	0.93	0.93	7590

- **K Nearest Neighbors:** K Nearest Neighbors (KNN) is a non-parametric classification algorithm that works by finding the K nearest data points in the feature space to a given sample and assigning it to the most common class among those neighbors. The choice of K is important and has been set to 5 in our case - as we have 5 buckets to classify the tweets into.

	precision	recall	f1-score	support
religion	0.92	0.46	0.61	1587
age	0.9	0.75	0.82	1574
gender	0.76	0.8	0.78	1490
ethnicity	0.89	0.89	0.89	1547
not_hatespeech	0.45	0.77	0.57	1392
accuracy			0.73	7590
macro avg	0.78	0.73	0.73	7590
weighted avg	0.79	0.73	0.74	7590

- **Logistic Regression:** Logistic Regression is a classification algorithm that models the probability of a binary or multi-class outcome based on the input features. It works by fitting a logistic function to the data that maps the input features to the probability of the outcome. Logistic Regression is simple to implement and can handle both continuous and categorical data.

	precision	recall	f1-score	support
religion	0.96	0.96	0.96	1587
age	0.96	0.95	0.95	1574
gender	0.95	0.86	0.9	1490
ethnicity	0.98	0.97	0.98	1547
not_hatespeech	0.77	0.87	0.82	1392
accuracy			0.92	7290
macro avg	0.92	0.92	0.92	7590
weighted avg	0.93	0.92	0.92	7590

For this project, the default hyperparameters were used for the models. An attempt was made to use GridSearchCV to increase the accuracy with Random Forests, but no significant change was seen. This might also imply that the Random Forest's default parameters were optimum for the dataset already.

Thus, it can be concluded that the Random Forest Model was the best choice for classification, with a classification accuracy of 94%.

8.2 Deep Learning Approach

For the deep learning approach, a LSTM (Long-Short Term Memory Network) was used. It is a Specialized type of RNN that can learn long-term dependencies between data. LSTMs have been shown to perform well in NLP based classification tasks. Their ability to model complex relationships and patterns in the data can help to improve the accuracy and robustness of the classifier.

For the classification, the following steps are involved in training the LSTM on the data, and carrying out tests.

Preprocessing data

Cleaning: For preprocessing the data, the text that was cleaned in the first part was used as it is. That is, the text was processed by cleaning, lemmatizing, stemming, and removing stopwords.

Tokenization: Splitting the text into individual words or tokens. This can be done using whitespace or other delimiters such as punctuation marks.

Padding and truncation: Ensuring all input sequences have the same length by padding shorter sequences with zeros or truncating longer sequences. Padding involves adding zeros to the beginning or end of sequences that are shorter than a specified length, until they reach the desired length. Truncation involves removing words from sequences that are longer than a specified length, so that they also meet the desired length. This can be done by discarding words from the beginning or end of the sequence.

For example, consider the following sequence of words: "I am happy". If we set the maximum sequence length to 10, we can pad it with zeros to get a length of 10: "I am happy" becomes [0 0 0 0 1 2 3 4 5], where the numbers correspond to the word indices in a dictionary.

Vectorization: Converting the tokenized words into numerical vectors that can be fed into the LSTM. This is typically done using word embeddings.

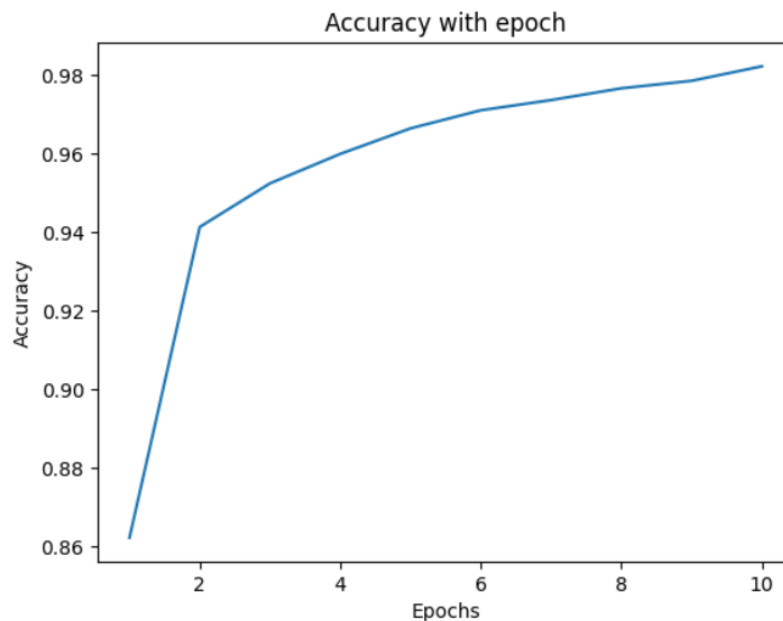
Implementation

For the LSTM, the following parameters were used for training. Their significance as well as their values have been attached.

- **vocabulary_size:** This is the maximum number of words to keep based on word frequency. It is set to a specific value to limit the number of unique words in the input data, which can help improve the efficiency of the model. The vocabulary size was 5000.
- **input_length:** This is the length of each input sequence. It needs to be set to a fixed value to ensure all inputs are the same length and can be processed by the model. It corresponded to the maximum number of allowed words in each tweet, and was set to 120.
- **Embedding layer:** The embedding layer converts words into fixed-size vectors that can be processed by the neural network. The Embedding() function takes in the vocabulary size,

the size of the output vectors (128 in this case), and the input length.

- **SpatialDropout1D:** This layer applies dropout to the embeddings. The 0.4 value indicates that 40% of the elements of the input will be randomly set to 0 during training, which helps prevent overfitting.
- **LSTM:** This layer contains the LSTM cells that process the sequence data. The 256 value is the number of LSTM cells in the layer, and the dropout and recurrent_dropout values (both set to 0.2) indicate the fraction of the input units to drop during training.
- **Dense:** This layer is the output layer of the model, which contains five nodes (one for each category of hate speech). The softmax activation function is used to ensure the output probabilities sum to 1.
- **batch_size:** This is the number of samples processed before the model is updated. The larger the batch size, the faster the training time, but also the more memory required. In this case, the value is set to 64.
- **epochs:** This is the number of times the entire training dataset is passed through the neural network. In this case, the value is set to 10.



The above graph depicts the increase in the accuracy of the sample with each epoch. We can see that the accuracy was somewhat saturated after 6 epochs, and crossed the maximum efficiency acquired by ML models (94%) by the second epoch.

Thus, the hate speech classifier was developed using ML and Deep Learning. The LSTM fared much better than other methods and can be used as an effective hate speech classification model.

9 Future plans (and some incomplete ideas)

An attempt was made to make a similar Hate Speech Classifier in an Indian Context. The popular HASOC dataset, which has hate speech from twitter from Indian Media sources, politicians and other users was used for the same. The machine learning algorithms were used, but a very low accuracy in the range of 57% (using both SVM and Logistic Regression) was obtained. This was likely due to the small number of data points for the dataset. LSTM wasn't run for this dataset yet.

10 Part 2: Analysing PM Modi's Tweets

Overview

This project aims to analyze tweets by Prime Minister Narendra Modi to identify any interesting patterns in the data. The tweets were scraped using the `sncscrape` library, and a dataset of tweets was produced, with tweets ranging from 2009 to 2022, with a total of almost 19000 tweets. The tweets were preprocessed using various techniques discussed in depth in Part 1, and the `Vader` library was used for sentiment analysis on the tweets.

The sentiment analysis results showed that the majority of the tweets were neutral, while a smaller proportion of tweets had a positive sentiment. The topics of the tweets were also analyzed, and it was found that the majority of the tweets were related to politics, governance, and development.

Overall, this project provides insights into the topics and sentiment of tweets by the Prime Minister, and highlights his focus on development and progress. PM Modi's Twitter account, possessing the most followed Twitter account in India, acts as a significant part of India's social network. The findings of this study can be useful for policymakers, researchers, and social media analysts who are interested in understanding the sentiments and topics of political tweets.

Scraping and cleaning Tweets:

For the project, the tweets were scraped in Python using `SNScrape`, an open-source and collaborative web-crawling framework written in Python. It is used for extracting the data from websites, as per our requirement. Specifically for Twitter, it allows for the scraping of tweets, retweets, replies, and other information such as user profiles, hashtags, and geotags.

The process of scraping tweets using `SNScrape` was still quite tedious, as it tended to fail quite often, owing to the Twitter API's rate limits on bots. Thus, the scraping was done in small batches of 1000 tweets, and the tweets were stored in separate csv files that were later merged into one csv file. However, there was no alternative, as `Tweepy` had even stringent rate limits, with the reading tweets functionality being removed recently, making it impossible to access tweets using `Tweepy`.

Ideally, `SNScrape` should have been able to scrape all tweets, but owing to some very recent changes in the Twitter API, we were unable to scrape any additional tweets. Thus, we stopped our search after 19000 tweets.

The procured data was cleaned using the same techniques as used for the hate speech classifier, i.e. Removing Null values, stemming, lemmatization, removal of stopwords etc.

It was observed that there were some tweets (about 400 out of 19000) that were in regional languages. These tweets were not considered for sentiment analysis, but were considered for the analysis elsewhere. These were eliminated from the dataframe.



Figure: Word clouds with positive sentiment

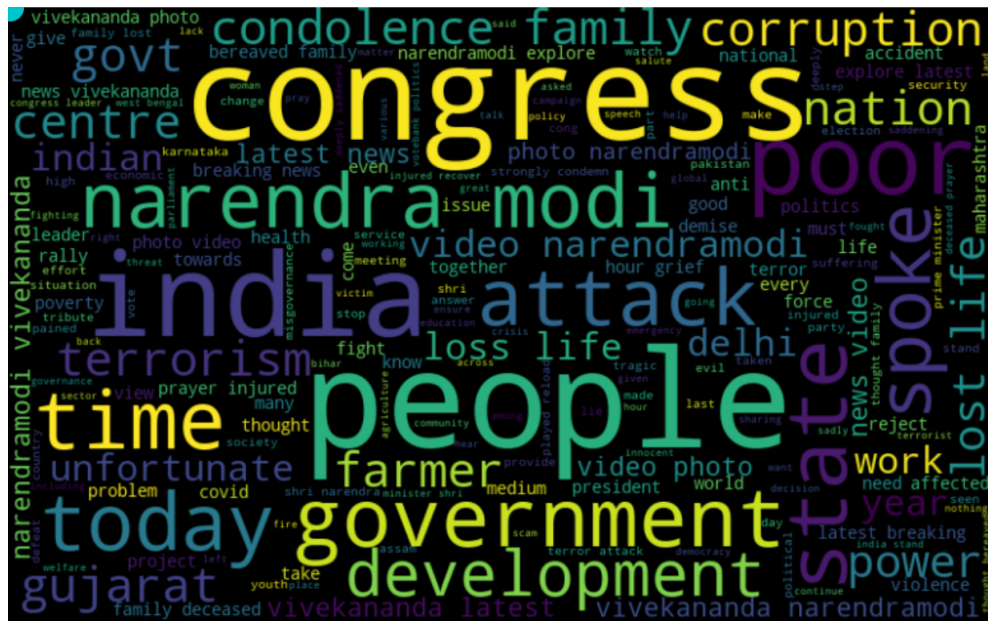


Figure: Word clouds with negative sentiment

As we can observe, the word clouds describe the general sentiment in Modi's tweets: The word cloud of the most common words includes "Visit", "Thank", "Gujarat", "Congratulate", "Congress" etc, which shows us the primary intents of the twitter handle to inform the people regarding the visits of any foreign dignitaries (or PM Modi himself), congratulating, and occasionally lambasting the congress.

The positive sentiment word cloud shows words like “Mannkibaat”, “salute”, “development”, “support”, which are expected - positive sentiments are associated with the Army, development, government’s initiatives etc.

The negative sentiment word cloud, on the other hand, includes “farmer”, “congress”, “poor”, “condolence”, “corruption” “loss” etc, implying that the majority of the negative tweets are regarding poverty, any disaster, policy changes, terrorism and the Congress.

Plotting the Number of tweets per day

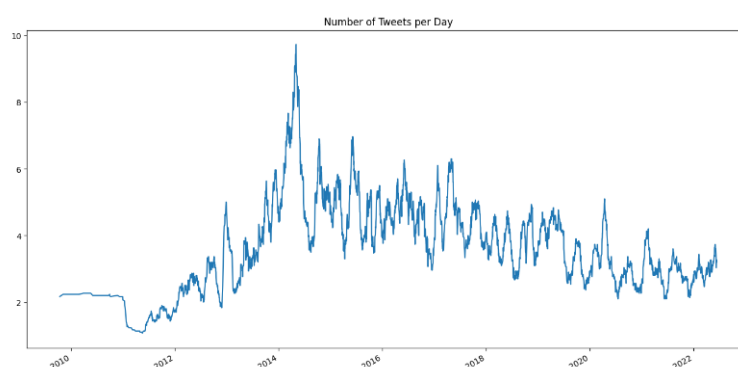


Figure: Plot for number of Tweets per day

Plotting the Number of tweets per day, we see a few major spikes - one during 2014, one during 2016, a consistent plateau during 2019 and during 2020. These correspond to some of the biggest events during Modi’s tenure.

In 2014, the General elections were conducted in India during the months of July- August, which is when we see a large spike. Thus the tweets are meant for campaigning.

In 2016, the Modi Government announced demonetization - that is, notes of Rs. 500 and 1000 were declared invalid. This, again, is responsible for a large rise in the number of tweets, to provide verifiable news regarding the policy, and to curb any sources of fake news.

In 2019, again, the General Elections were to be conducted, and we see a consistent plateau during this period. This period corresponds to an increase in the number of tweets, which are again meant for campaigning purposes.

The spike in 2020 corresponds to the COVID-19 crisis in India, which is when we saw a large increase in the number of cases, with curfews and lockdown being enforced across the country.

It can be concluded that that Twitter handle of PM Modi sees changes in number of tweets during two events: Elections, and times of crisis, when news becomes the most

valuable resource.

Do note that the rolling average of the tweets of every month was shown here. This was done to slightly smoothen the data, to make it easier to visualize, and remove any noise. This has been done for all the graphs below.

Plotting likes and retweets with time

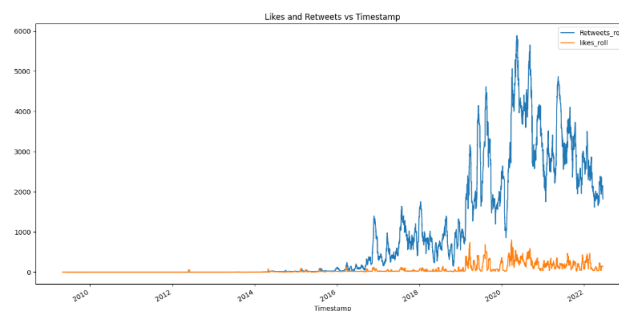


Figure: Plot of likes and retweets

As we can see here, there is a major discrepancy between likes and retweets. We are not sure if this is something that is associated with twitter's general usage, but we saw a very large discrepancy between the number of likes and the number of retweets. In general, the number of retweets was three to four times more than the number of likes, with the ratio being much more skewed elsewhere.

Plotting sentiment score vs time

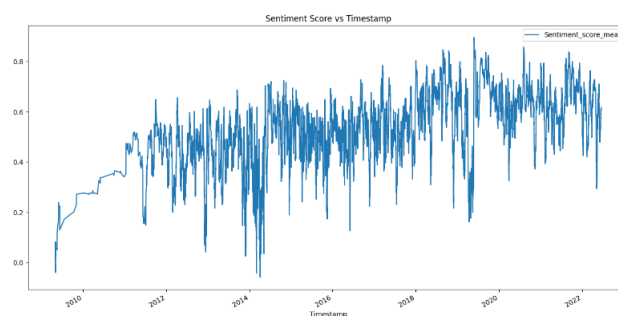


Figure: Plot of sentiment score vs time

We additionally plotted the sentiment score of the tweets with time, as can be seen here. We can observe that the sentiment score of the tweets significantly rose after 2015, after PM

Modi became Prime Minister. We can also observe that it dipped significantly during 2014, that is, during the election period. This dip implies that the sentiment of PM Modi's tweets during the elections was not positive in general - the tweets were negative, and generally targeted towards the Congress. This can also be confirmed making further analysis on the data. A similar dip was also observed during 2019, during the election period. Some smaller dips in the sentiment can also be observed during 2020, during the COVID 19 crisis. It can be seen that the sentiment is slowly getting more positive with time, however.

From this data, a very strong correlation can be drawing between election periods and the daily number and sentiment of the tweets.

Plotting Average engagement for each sentiment

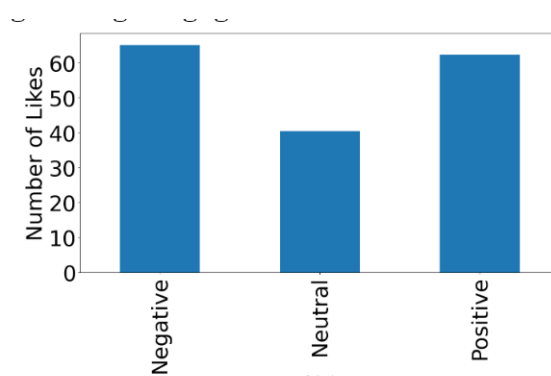


Figure: Average number of Likes of each sentiment.

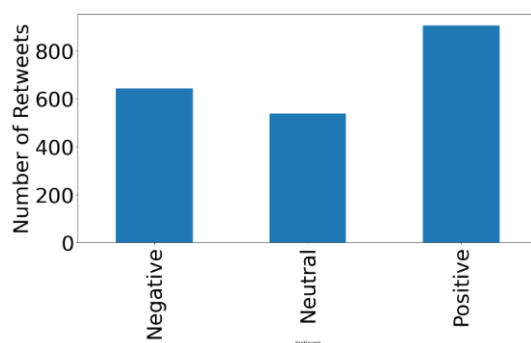


Figure: Average number of Retweets of each sentiment

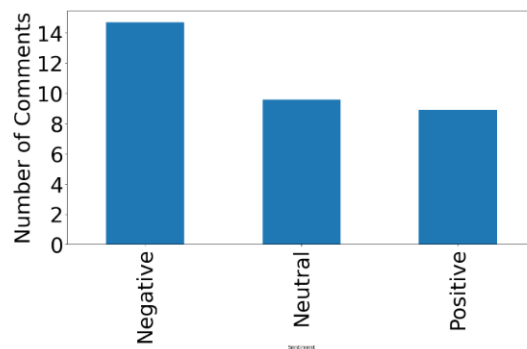


Figure: Average number of Comments for each sentiment

Yet another interesting graph is that of the likes per sentiment category. If we observe the average number of likes and retweets for each sentiment category, we can observe that it is the most polarizing comments, i.e. the ones that are mostly positive or negative, that get the most attention. Regarding the comments, which are the most important metric of engagement, it is seen to be high generally with negative sentiments.

This not only tells us about people's engagement with PM Modi's tweets, but about the Twitter algorithm in general - The polarizing tweets get more engagement, particularly comments for negative tweets, which results in the Twitter algorithm boosting its reach, which further results in more engagement from people, resulting in a vicious circle that promotes polarizing thoughts.

11 Links for notebooks and datasets

Visit the GitHub Repository :

<https://github.com/yashkokane291/CS328-Datasets-and-notebooks.git>.