Homework - 3

Isha Goel (2017054)
Ashwin Singh (2017222)

Rumours in Delhi Riots (Jaffrabad)

24th March 2020

Data Collection

We used **Twint -** a twitter scraping tool to collect our data.

We included the following list of police handles: @DelhiPolice, @DCPSouthDelhi, @CPDelhi, @DCPNewDelhi, @DcpNorthDelhi, @DCPCentralDelhi, @DCPNWestDelhi, @DCPNEastDelhi, @dcp_outernorth, @DCPEastDelhi, @DCPSEastDelhi, @dcp_southwest and DCPWestDelhi.

We also included multiple spellings of Jaffrabad (Jafrabad, Jaafrabad etc.) to compensate for spelling errors that the users might make in their tweets.

Tweets were collected using queries of the form `@DelhiPoliceAccount Jaffrabad` to retrieve tweets containing the keyword "Jaffrabad" and @DelhiPoliceAccount in the form of a mention. Additionally, we collected information about the users who created the tweets, such as their bio, number of followers, following, whether they're verified, age of their accounts, etc.

All tweets were annotated into 2 types of labels based on

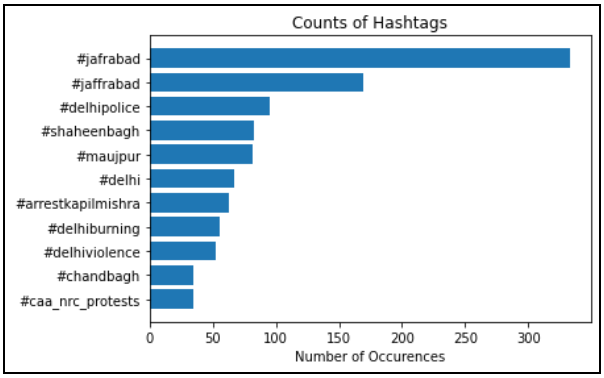
- **Police context** - Action/Help/Question/Praise/Report/Other
- **Information** - Fact/Rumour/Chaos

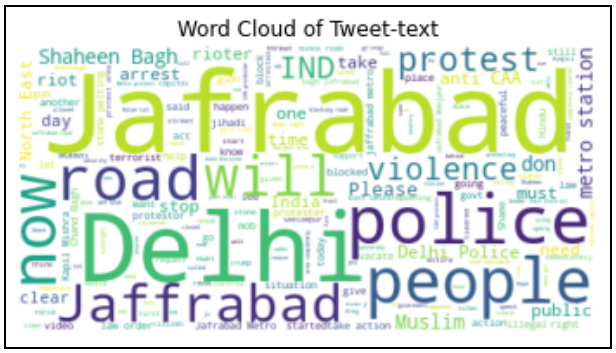
Preprocessing and Extraction

We used the following regular expressions to extract **PII** from textual data in the form of tweets and user bios:

External Links: `https\S+$`
Phone Number: `^[6-9]\d{9}$`
Vehicle Plate No: `^[A-Z]{2}[ -][0-9]{1,2}(?: [A-Z])?(?: [A-Z]*)? [0-9]{4}$`
Email ID: `^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$`

We also performed some **preliminary analysis** of the data and retrieved the following plots:



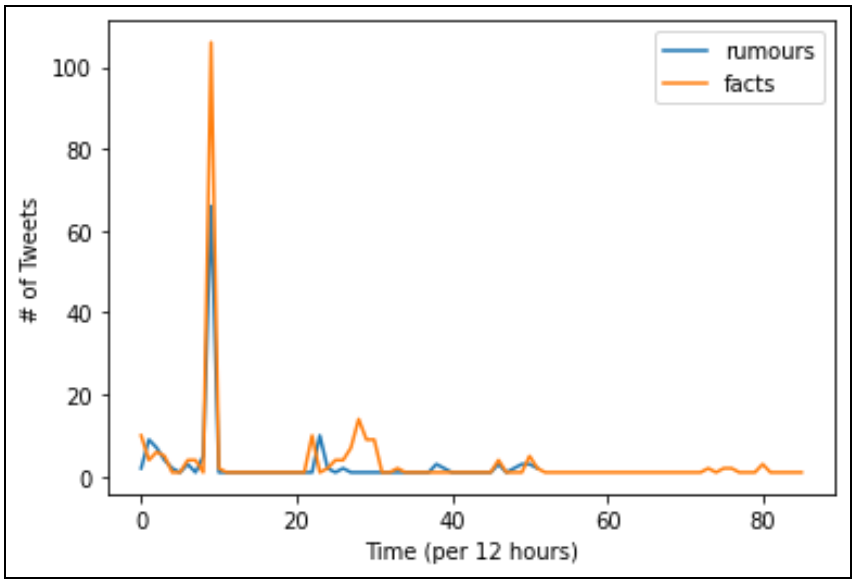


Contextually, #arrestkapilmishra and #caa_nrc_protests can be seen represent key information about the protests.

- **CAA-NRC** has been a burning topic of discussion across the country ever since the citizenship amendment bill has been passed and is the cause of these protests.
- **#ArrestKapilMishra** is based on an a set of events during the visit of President Donald Trump when some roads in Jaffrabad were blocked along with the metro-station, causing inconvenience to the residents of the area.

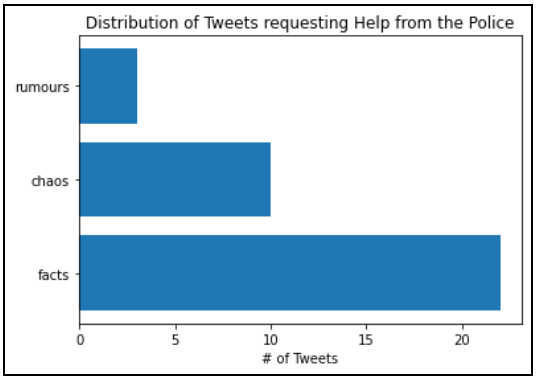
  Former MLA Kapil Mishra was blamed for inciting a pro-CAA mob in a speech where he offered an ultimatum to the police. More on the same can be found on this link. Most of our tweets containing rumours and news stories were based around these set of events.

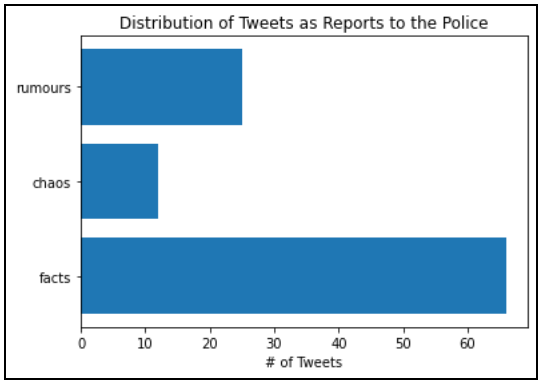
  The peak in the time-stamp graph (denoting the maximum # of tweets within time intervals of 12 hours) also refers to these set of events which were predominantly tweeted about across this time period.

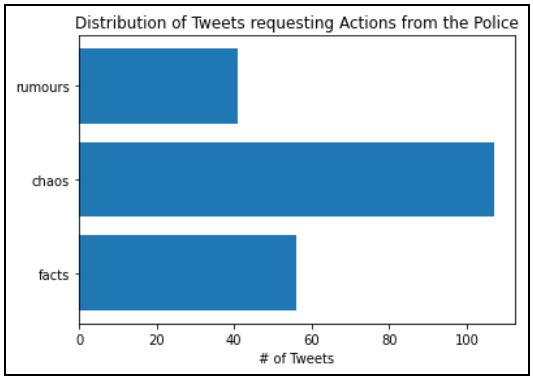


Upon plotting information from our annotated tweets, we also concluded the following:

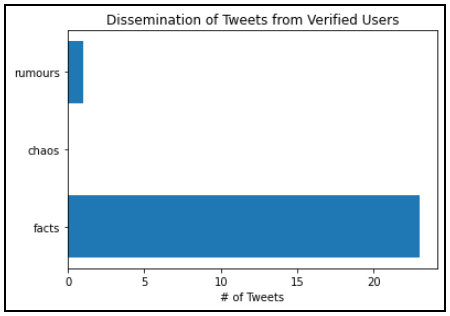
- Tweets requesting **help** from the police are mostly **facts**.
- Tweets demanding **action** from the police are mostly **chaotic**.
- Tweets in the form of **reports** to the police are mostly **facts**.

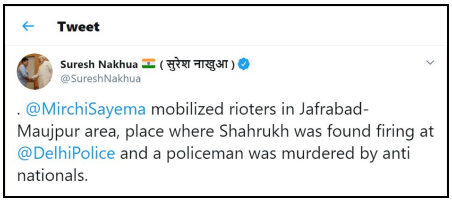




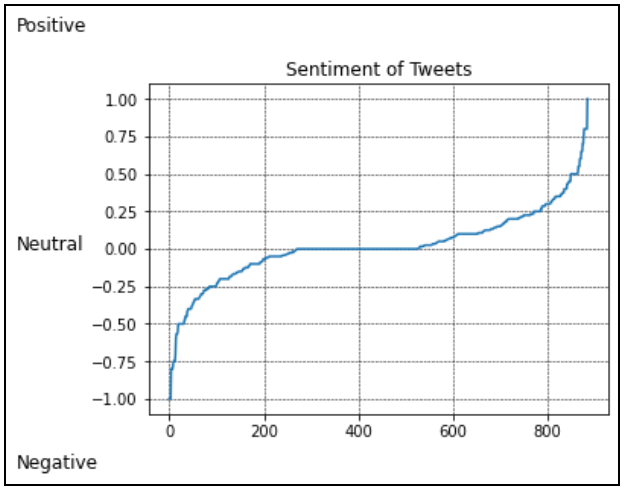


We also made a distribution table for the tweets of verified accounts and made an interesting observation - there was only one verified account which had tweeted a rumour.





(The verified profile and the rumour)



We were able to extract the following **features** from the data collected

Textual: Captions, Hashtags, User bios

Non-textual: Location of Account, Age of Account, # of Likes, Retweets, Replies on the tweet, # of Followers, Following of the User, Sentiment of the Tweets.

Model

Feature Selection: We considered 9 features for every tweet, namely number of followers, following, number of likes, retweets, whether they're verified, age of their account, the sentiment of the tweet text and the plabel that we assigned during annotation. Categorical feature **plabel** and class label **ilabel** were encoded using a **Label Encoder** to convert string type data to integer type data for use in the model.

Training, Testing: We deployed a Support Vector Machine (SVM) with an *RBF* kernel for training and classifying the tweets into 3 classes (information-based) - **facts, rumours** or **chaos**. Due to class imbalance, we used **Oversampling** to create synthetic training data and avoid bias towards a particular class. Both training and testing data were normalized prior to being used in the model.

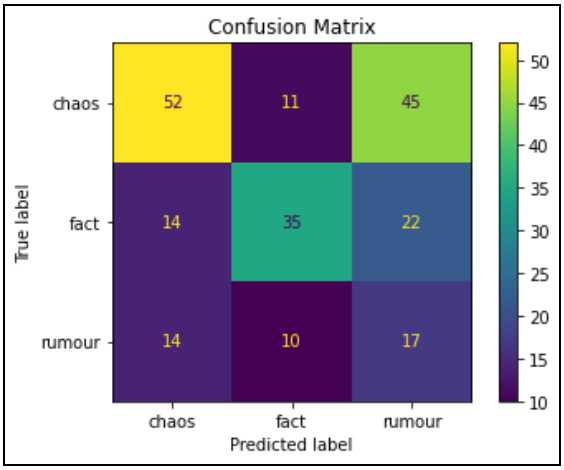
Efficacy of the Model

We ran our model on a test set of 220 tweets and got the following results:

	Precision	Recall	F1-score	Support
chaos	0.65	0.48	0.55	108
fact	0.62	0.49	0.55	71
rumour	0.20	0.41	0.27	41
Accuracy			0.47	220
Macro avg	0.49	0.46	0.46	220
Weighted avg	0.56	0.47	**0.50**	220

Our model yielded a weighted accuracy of 50%. We also observed the following:

- Among *false-negatives*, it classified more rumour-tweets as chaotic than factual which is less harmful i.e. chaos acts as a safe label.
- Among *false-positives*, a lot of chaotic tweets have been classified as rumours due to low inter-class variance.



Rumour-Spread Score

The SVM model yielded class probabilities for each tweet i.e. given a tweet, it was able to identify the probabilities associated with that tweet to belong to each of the three classes - fact, rumour and chaos. We utilized the class probability of rumour for each tweet and saved it in the data.

We considered different aspects of twitter user to formulate a score for rumour-spread.

User Attributes: Followers of the user who created the tweet, whether the user is verified or not. We applied *Pareto-principle* in weighing these factors - 80% of the content is generated by 20% of the users on social media, so we weigh the user's followers by a factor of **0.2** and amplify this to **0.6** if the user is verified on Twitter. So, we get the following measure:

m = (verified*0.4 + 0.2)*(# of followers) **where** verified = 1 if user is verified and 0 otherwise.

To convert the range of m to **[-1,1]**, we use a sigmoid function and transform this measure into the expression: $\mathbf{S_1 = 1 / (1 + e^{-m})}$

Activity Attributes: to measure the activity of followers of the user who created the tweet, in terms of likes, retweets, replies. Since a retweet remains on the user timeline, even after appearing on the user feed, we assign it twice the weightage relative to likes and replies, to generate the following measure.

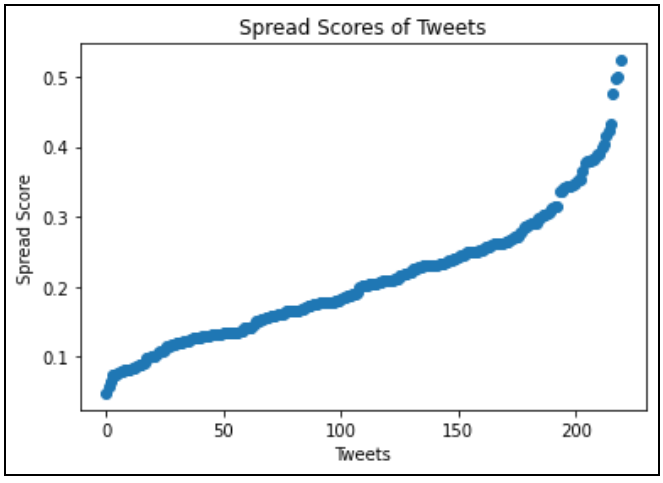
m = 0.25*likes + 0.25*replies + 0.5*retweets

Again, to convert the range of m to **[-1,1]**, we use a sigmoid function and transform this measure into the expression: $\mathbf{S_2 = 1 / (1 + e^{-m})}$

To calculate the rumour spread for a given tweet, we combine the two measures for it and weigh it by the rumour likelihood assigned by our model:

$$\textbf{rumour\_spread\_score} = \textbf{rumour\_likelihood}*\mathbf{S_1}*\mathbf{S_2}$$

Given below is the graph plotted for the rumour spread scores across tweets in the testing data -



Suggested Modifications to the Model

- **Dataset:** Current size of the dataset is roughly 885 tweets which isn't large enough to accurately classify the tweets into three different labels. The data also deals with class imbalance which often requires oversampling or undersampling while training the data. Collection of a larger and more diverse dataset would improve the model and make it more holistic.
- **Additional Features:** Textual features such as term frequency (TF), inverse document frequency (IDF) etc. can be extracted from tweets, bios and twitter feeds of users. Timestamp of tweets within the data can be leveraged to provide temporal information to enhance the model.
- **Aggregation of Data:** When the database of users becomes large enough, rumour-spread-score over multiple tweets of an account can help identify whether that account is 'suspicious' or not. We can also make networks of users as well as tweets to enhance rumour-spread-score and detect clusters of people spreading rumours or factual information. Within these clusters, we can discover strongly connected components and users who act as central hubs of information dissemination to find sources from where rumours emerge.
- **Long-term Training:** One could convert the model into a semi-supervised learning classifier by showing the user a dialog box for tweets which are ambiguous (have similar probabilities across classes - rumours, facts or chaos) so that the user can annotate them in real-time. Similarly, tweets predicted as facts/rumours with low confidence scores can be placed under review. Long-term training in such a manner would yield a more robust model which can detect rumours accurately.

Setting-Up Guide

For setting up real time classification of tweets we have made a terminal interface for the police to use which leverages our already trained model.

1. Ensure that all files - model **SVM.sav, police.py, requirements.txt** are present in the same folder. Within the same folder make a **credentials.py** file and save api_key, access_token under the following variable names:

   ```
   api_key, api_key_secret, access_token, access_token_secret
   ```

2. Within the same folder, open command prompt and install all requirements needed to run the classification script using the following command:

   ```
   pip install -r requirements.txt
   ```

3. Upon installation, run the classification script on the command line:

   ```
   python police.py
   ```

   The script would require the user to provide the link of the tweet and choose an appropriate category for it from **action / help / other / praise / question / report.**

   ```
   PS C:\Users\rimco\Desktop\PSOSM\Homework 3> python .\police.py
   Enter the link to Tweet: https://twitter.com/KapilMishra_IND/status/1242095477870325762
   How would you best describe the tweet? (action / help / other/ praise / question / report): report
   ```

4. Upon input of this information, the classifier script yields the following output as result:

   ```
   The Tweet is a rumour
   LIKELIHOOD
   CHAOS: 0.37693697095605094
   FACT: 0.22254804348595006
   RUMOUR: 0.4005149855579989
   ```

   Along with the predicted class, it also yields the likelihood of the tweet across all classes.