# Social Media (Twitter) Analysis Functionailty

The work under social media analysis consisted of 3 parts:

- **Data Collection** – Tweets and other twitter related data
- **Overlap Analysis** – Analysis of overlap between 7 national dailies under study and 2 national political parties.

The relevant scripts and other data are contained in Scripts.zip

Some terms

- Newspaper communities – The followers of the twitter handles of the 7 national dailies under study

## 1. Data Collection

Most of the data collection was done using Twitter REST API and its python implementation tweepy. To use these, authentication tokens and keys are required, which are given in SM/Res/twitter_access_keys.csv. The API(or tweepy) returns the tweets in the form of tweet/status objects which can be handled and stored as JSON objects. In general most of the datareturned by the API is in the for objects, which can be handled as json. While using the API(or tweepy), you may encounter a number of errors, for which twitter returns an error code. Almost all such cases are of rate limiting and unauthorized access. *Rate limting*, the calls(functions) provided by the API( or tweepy) are rate limited i.e. in a window of a certain time period, you can only make specified amout of calls. *Unauthorized access* mostly occurs when trying to access tweets of an user who requires you to follow them to check their tweets. Such errors need to be handled through exceptions.

### 1.1. Tweets

Tweets about the 4 policies were collected – GST, Demonetization, Aadhar and Farmers' Protests. Two types of tweets were to be collected:

i) Tweets made by newspaper community of the dailies under study (called *community tweets*)

ii) Tweets sharing the article urls of policies under study (called *url tweets*)

#### 1.1.1. Community tweets were dowloaded using the following method:

Given a twitter userid we can download upto about 3000 latest tweets for the user using the tweepy function, *user_timeline*. The script mpd.py in SM/SM_Scripts downloads the tweets in the above manner for a list users given in a json file. The script tries to download tweets of an user, in case of exceptions it logs them into log files dlog_file_n, where n is the process number. The script spawns multiple processes to maximize the utilization of internet bandwidth. Downloaded tweets of an user are stored in a JSON file named after the userid inside a folder dstoragen (n again is the process number) for further processing. For example, the script will generate a file 18839785.json for user id 18839785. Mpd.py requires two other python scripts te.py which are in SM/SM_Scripts (sets up the access tokens and keys) and trim_tweet.py (to only retain required field in the tweet JSON object).

To run mpd.py:

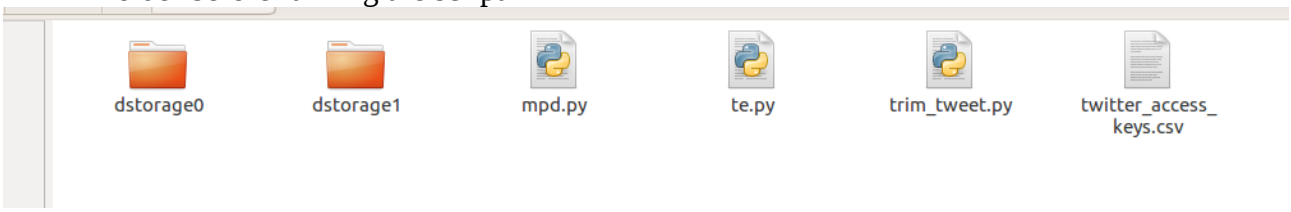
- Make sure the directory containing the script has te.py,trim_tweet.py and twitter_access_keys.csv.
- Make sure the directory containing the script has the *dstoragen* folder (starting from dstorage0 to dstoragen-1, where n is the no. of processes spawned)
- Run command: **python mpd.py <user list in a json file> <no. of processes >**

The <user list in a json file>, is the list of user-ids stored in python dictionary saved as a JSON file, dictionary keys being the user-ids. <no. of processes> is the no. of processes you want to spawn based on your net capabilities.

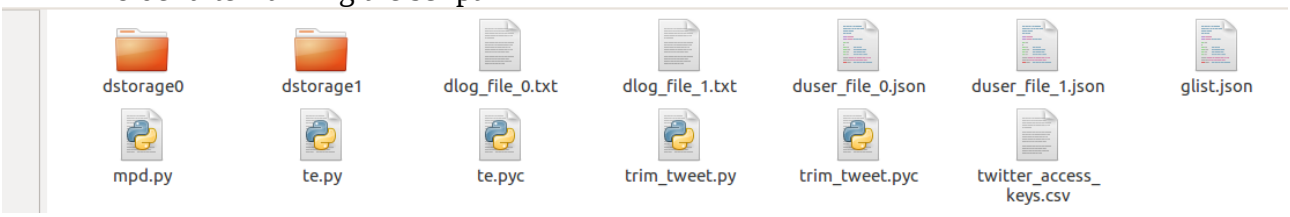
Example with images

Example JSON userfile

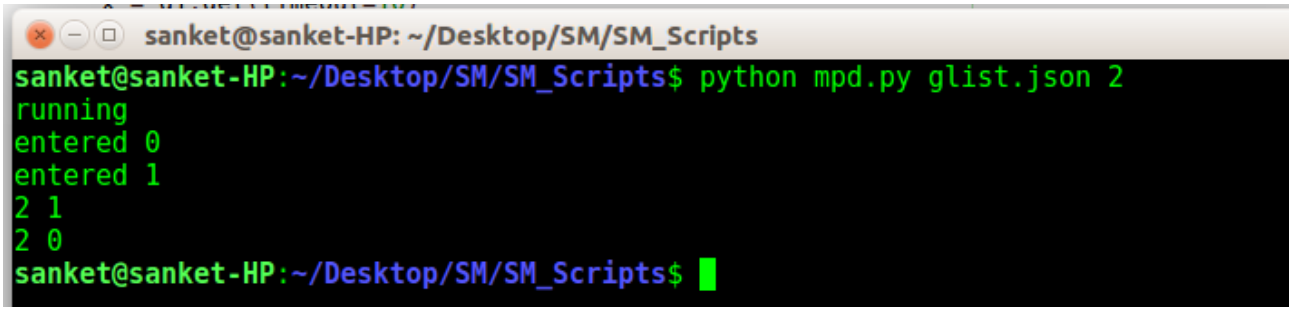

Folder before running the script



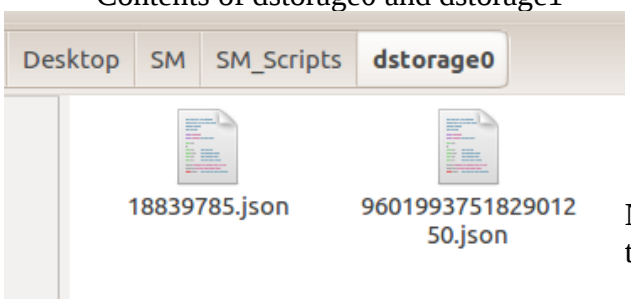

Folder after running the script



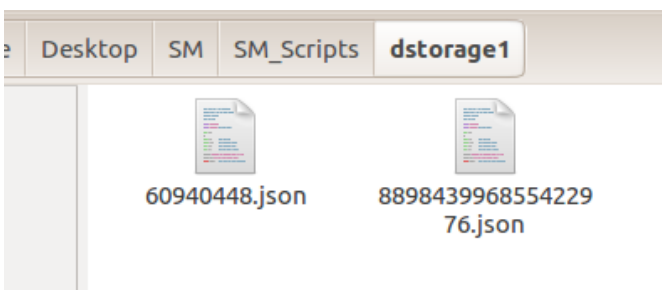

Terminal




Contents of dstorage0 and dstorage1




Next task



is to filter the downloaded tweets of each user for the given policy (GST,Aadhar,Demonetization and farmers' protest). This done by 1. searching the tweet text for existence of specific keywords of that policy, 2. checking the hashtags of the tweet. The script SM/SM_script/filter.py is used for this. The script has two functions is_keyword_aad() and is_keyword_demon() for filtering tweets about Aadhar and Demonetization respectively. Other functions should be created based on the process above for other policies that may be studied in future.

To run it : **python filter.py <arg1> <arg2>.**
arg1 is a line separeted file containing address of the JSON files created by mpd.py
arg2 is the output filenameof the JSON file where you want the filtered tweets to be.

The keyword list for the topics are as follows:

- Aadhar - ["aadhar", "adhar", "aadhaar", "adhaar" ]
- GST - ["gst"]
- Demonetization - ["demonetization", "demonetisation"]
- Farmers' protest - ["farm loan","crop loan", "farmer suicide", "debt waiver", "waiver scheme", "farming community", "farmer agitation", "plight farmer", "distressed farmer", "farmer issue", "farmers protest", "agrarian crisis", "agrarian unrest", "farmers protests", "loan waivers", "agriculture protest", "farmer march"]

To capture more tweets about farmers' protest, I first filtered tweets for ["farm", "agrarian", "agri"] and then furthered the resulting tweets on ["loan", "suicide", "debt", "waiver", "agitation", "plight", "distress", "protest", "crisis", "march"]

### 1.1.2. URL Tweets

Articles published online by the newspapers are scraped ny the media team. These articles are shared on social media (twitter). These URL tweets are needed to be collected for various studies by the team. This done by scraping the *twitter advanced search* site. The site provides many features to search tweets, one of them being searching by url shared in the tweet text. To search on the site, we need to formulate a search query. For our purpose the query will be **url:<URL>** , where <URL> is the url for which you are searching the tweets for. The scraper used for this is publically available on the internet. It can be found at SM/SM_Scripts/Scraper. The script used to download the url tweets is SM/SM_Scripts/Scraper/Main.py. This can be modified to suit your requirements. To run the script :

**python Main.py <path of the file containing article urls> <output JSON file>**

The format of the file containing the urls is as follows:

.....
ith url;newspaper name ny which the article is paublished
.....

Example

```
4  http://www.thehindu.com/todays-paper/tp-national/tp-karnataka/
   mills-in-mandya-to-crush-cane-from-august-1/article7475833.ece;HINDU
5  http://www.thehindu.com/todays-paper/tp-national/tp-kerala/
   families-unite-to-reap-success-in-organic-farming-in-idukki/article5608111.ece;HINDU
```

### 1.2. Follower List

Followers lists of the 8 national dailies and two political parties were required for overlap analysis. These can be found at https://drive.google.com/open?id=1vX8cWJlLCeJXBaTOf_APoacP6j-4r5gX. The follower list of any account can be downloaded using the tweepy function *follower_ids*, this function takes user id of an twiiter handle and returns the user ids of followers the account. The script SM/SM_Script/follower.py does this. To run the script: **python follower.py <outfile path> <account id>**, where <account id> is the user id of the twitter handle whose follower list you want to obtain. The script produces 3 files : output file, an error log named 'error_file' and a 'cursor_file' to keep track of cursors in case of unexpected process termination. This helps in cases when a significant amount of follower list has been downloaded. Keeping the track of cursors allows us to start the download from where it stopped.

## 2. Overlap Analysis

Note: Only pairwise overlap has been considered.
The objective of overlap analysis was to:

- Measure the extent of overlap in online news communities.
- Find and verify if newspaper communities with larger extent of overlap (as compared to others), exhibit similar sentiment on social media (twitter).
- Does the overlap in online news communities correlate with the sentiment expressed by the newspapers in their articles.

### 2.1. Measuring the extent of overlap

The extent of overlap can be measured using Jaccard's coefficient. For 2 sets A and B, Jaccard's Coefficient is calculated using $(A \cap B)/(A \cup B)$.

Another way to measure this is Odds Ratio. The odds ratio for a pair of newspaper (i,j) is calculated as follows:
Consider the total no. of users in the newspaper communities irrespective of what newpaper they follow. This is our universal set. Therefore, the odds ratio is the ratio of the (odds of an user from the universal set to follow newspaper i given it follows newspaper j) and ( odds of an user from the universal set to not follow newspaper i given it follows newspaper j). Lets represent this as OR(i,j). It is observed that OR(i,j) = OR(j,i). This can also be visualized as a graph, with nodes representing newspapers and edges between the nodes being the OR values.

The calculation and results for OR and JC values can be found in SM/Res/CR folder. The OR graph visualization is present as SM/Res/Plots as odds_ratio_graph.png.

### 2.2. Sentiment Analysis

To analyse the similarity between the sentiments of different news communities, we need to first evaluate the sentiment of each tweet. This done by using VADER sentiment analysis tool. Its python implementation is used in the script SM/SM_Script/ljr.py to add sentiment fields to the tweet JSON object. To run the script: **python ljr.py <arg1>**, where <arg1> is a line separated file containing the list of JSON files containing tweets without sentiment values. This script produces new JSON files with tweet objects having

sentiment values as added fields. The sentiment values produced by vader range from -1(negative) to +1(positive)

To further use the sentiment data for analysing the similarity (or difference), plot the graph of CDF (of tweets) vs sentiment for different news communities. Similar or close CDF plots imply similar kind of sentiment expressed by the community. From this we can check if communities with stronger overlap have similar (or closer) CDFs.

All the CDF plots can be found in SM/Res/Plots folder.