Sentiment Analysis and Community Detection in Twitter

Network Data Analytics

Niranjankumar Gangaraju
Faculty of Business and Information Technology
Ontario Tech University
Oshawa Ontario Canada
niranjankumar.gangaraju@ontariotechu.net

Sai Sandeep Borra
Faculty of Business and Information Technology
Ontario Tech University
Oshawa Ontario Canada
saisandeep.borra@ontariotechu.net

ABSTRACT

Twitter is the 3rd most used social media platform with greater than 340 million current users and generates a mean of 500 million tweets a day on trending topics. This huge data produced, and the twitter API availability provides us a perfect opportunity for twitter data mining and performs various other activities like sentiment analysis and community detection. There is a huge debate on Sentiment and mining opinion of social media for research. Be that as it may, the greater part of the cutting edge works and inquiries about on the programmed sentiment analysis gathered from social media and microblogging sites by classification into "+ve" and "-ve" or classification into "+ve," "-ve," and "balanced" of texts. Right now, the project we are implementing a methodology that, classifies the tweets into positive negative and neutral. Along with this we are implementing community detection for a single user. Edge weight is the represented by number of times the used has re-tweeted or replied the user and node diameter is determined by the number of outgoing links. For this project we are using Jupyter notebook for getting the refining data for sentiment analysis, getting the user data for community detection and Gephi for plotting and finding out the various communities for the given user.

KEYWORDS

Twitter, Sentiment Analysis, Community Detection, Tweepy, binary classification.

INTRODUCTION

Twitter is one of the fastest growing microblogging sites. The growth in the users is quite exponential and it is predicted that microblogging sites like twitter are going to take over the news industry. Twitters API connectivity provides a unique feature to access the live tweet data for business purposes. Businesses uses this data to provide better products, services and options to the customers based on the tweets. To get access to the twitter data developer need to get access to twitter developers account by filling out a questionnaire. The process takes about one week to get the access. Once the develop gets the access he needs to create an app by filling out the form, naming the app and informing the purpose of the app. Each app has its own unique app id. Once the app is created in the developers account, we can access the required keys to connect own developers' environment to twitter API.

Sentiment analysis is the knowledge and categorization of feelings (positive, negative and impartial) inside test information utilizing text inquiry techniques. Sentiment analysis permits organizations to distinguish client conclusion toward items, brands or administrations in online discussions and input. Sentiment analysis in twitter is the interpretation of tweets to find the emotion hidden inside the tweet and check if the tweet is positive or negative. For conducting sentiment analysis, we are writing our code in Jupyter notebook. We are importing tweepy, pandas, re, nltk, matplotlib, collections and itertools python libraries to perform sentiment analysis.

For sentiment analysis, we are going to get the tweets using python tweepy library that is with a hashtag (#). Then we perform data cleaning process by using nltk library we are going to remove the stop words, remove the mentions (@), by using re library we remove the URL if mentioned in the tweets and we have to convert the all the tweets to lowercase along with splitting each an every word in a tweet. Later we can find the most common words in all the tweets in a bar graph. By using python textblob library we can get the polarity (which ranges from -1 to 1) of the tweet by which we can plot a histogram of all the tweets based on polarity.

For Community detection like the above case, we get the tweets by using the python tweepy library. We are only considering 1-step Neighbourhood that is we consider only 2 levels of friends with a condition that we will not consider friends of friends who are not friends with the main user. We are creating a directed graph with edge weight and node weight. After this, we can create a network of Twitter users. Then we detected communities based on information we gather from python code and pass that information into node and edge csv file. These files can then be imported into Gephi visualization software for detecting communities in the network.

In order to get the datasets, we are using a custom application to connect to twitter API. Then by using the appropriate hashtags or user ID, we can get all the tweets and user-related information. The only issues with the data are that we must clean the unstructured data before we perform any visualization.

We have also implemented various visualization techniques like word cloud, histograms, bar graphs, community detection graphs, directed and undirected graphs.

LITURATURE SURVEY ON RELATED WORKS

A. SENTIMENT ANALYSIS

Sentiment analysis is the understanding and classification of feelings (positive, negative and impartial) inside test data utilizing text investigation techniques. Sentiment analysis permits organizations to distinguish client conclusion toward items, brands or administrations in online discussions and input [5].

Given a group of tweets, characterize each of them to one among the ensuing 7 categories: "love", "joy", "fun", "impartial", "detest", "misery" and "outrage". Extract various sets of features from tweets and to implement the classification machine learning algorithms is used. The sentiment classes for various reasons are chosen. To start with, it's presumed that a decent measure of data among negative and positive classes is required. Moreover, while the slants are those present the preeminent in tweets. Humans tend to act differently under various emotional situations. This incorporates the way they talk and express their sentiments. Consequently, it is essential to depend, on the vocabularies utilized, yet in addition on the articulations and sentence figures utilized under the various situations, to evaluate and demonstrate these emotions [5].

As expressed above, assessment-based highlights are ones bolstered the estimation extremity (i.e., "+ve" and "-ve") of the different parts of tweets. Concentrate just the resulting ones:

1. The quantity of +ve words which of -ve words.
2. The quantity of exceptionally passionate +ve words which of profoundly enthusiastic -ve words.
3. The proportion of passionate words.
4. The quantity of +ve and -ve emojis.
5. The quantity of +ve and -ve words [5].

Supposition investigation is that the procedure of naturally recognizing whether a book fragment contains passionate or stubborn substance, and it can besides decide the content's extremity. Twitter assumption classification expects to order the feeling extremity of a tweet as +ve, -ve or nonpartisan [4].

Texts are typically comprising of boisterous and ineffectively organized articulations and not ordinary articulations, not all-around shaped words and not the word reference terms. Before include determination, a development of pre-preparing (e.g., evacuating stop words, expelling URLs, supplanting refutations) are applied to downsize the amount of commotion inside the tweets. Pre-handling is performed widely in existing methodologies, particularly in AI based methodologies. In any case, hardly any investigations represent considerable authority in the impact of pre-preparing technique on the exhibition of Twitter notion examination [4].

To survey the impact of fluctuated pre-preparing strategy, six preprocessing strategies are applied to feeling classification utilizing four distinctive classifiers on five Twitter informational indexes. the whole examination arrangement comprises as follows. The pre-preparing strategies that are evaluated are as per the following [4]:

(1) Replacing negative notices. Tweets joins different ideas of refutation. When all is said in done, refutation assumes a significant job in deciding the estimation of the tweet. Here, the strategy for refutation is changing "won't", "can't", and "n't" into "won't", "can't", and "not", individually.

(2) Removing URL interfaces inside the corpus. Most scientists consider that URLs don't convey a lot of data with respect to the estimation of the tweet. Twitter's unexpanded URLs are extended to URLs and are then tokenized. At that point, the URL coordinating the tokens are off from texts to filter the tweet text.

(3) Reverting words that may contain copy letters to their standard English pace. Words with rehashed letters, e.g., "coooool", are normal in tweets, and people will in general utilize this design to exact their slants. Here, an arrangement of very three comparable alphabets is supplanted by 3 characters. for instance, "cooooooool" is supplanted by "cooool". Utilizing 3 alphabets recognize words like "cool" from "cooooooool".

(4) Deleting integers. By and large, integers are of no utilization when estimating notion and are detached from texts to filter the tweet message.

(5) Deleting stop words. Stop words typically observe the chief regular words in a very language, like "the", "is", and "at". Most scientists accept that stop words assume a non-positive job inside the capacity of assumption classification, which are being expelled before highlight choice by the analysts. The exemplary technique for evacuating stop words is that the procedure upheld precompiled records. Different records exist inside the writing.

(6) Enlarging abbreviations to their underlying words by utilizing an abbreviation library. Abbreviations and slang are exceptionally visit in tweets yet are not very much shaped words. It's expected to broaden them to their underlying words. Every abbreviation compares to a proof. Model, "∗4u" is "Kissforyou", "2 mro" is "tomorrow'.

An expanding measure of research has been designated for exploring the arrangement of strategies that exists to recognize good and ominous slants towards specific subject posted in semantic correspondence writings. The arrangement of uses for such investigation are various and fluctuated, beginning from newsgroup flame filtering and enlightening enlargement of modified reactions to examination of conviction and client input. for a few of those errands, ordering the client assessments into positive or negative is a urgent advance. Assessment examination might be given a role as a classification system, where the errand is to characterize messages into two classes figuring on whether they pass on positive or negative sentiments. Mining client sentiments and audits from online life storehouse is certainly not a straight assignment [1].

As the use of internet-based life has been expanded with a high rate, the data which is shared via web-based networking media by the clients is investigated to ask the information about the individuals. Numerous utilizations of notion investigation are actualized to comprehend the different notions of the clients. Individuals share

their status via web-based networking media which has data about their joblessness by posting articulation inside the assortment of negative sentiments [1].

Conclusion examination is an assessment of the assessment of the speaker, author or other subject concerning some theme. For instance, In US presidential political race 2016, Donald Trump, Hillary Clinton and Bernie Sanders were among the most noteworthy political race up-and-comers. The assessment of the overall population for an up-and-comer will affect the potential head of the nation. Twitter is utilized to store up a larger than usual various informational index speaking to the current general assessments of the applicants. The gathered tweets are breaking down utilizing dictionary-based way to deal with work out the sentiments of open. during this paper, we decide the extremity and subjectivity measures for the gathered tweets that help in understanding the client feeling for a competitor. Further, an examination is shaped among the up-and-comers over the sort of feeling. Additionally, a word cloud is plotted speaking to frequently showing up words inside the tweets [2].

A thought or view upheld feeling is generally expressed as a notion. Examining these passionate conclusions from various gatherings of clients and deciding their mentality towards the entire logical extremity or enthusiastic reaction to an archive, correspondence, or occasion is finished in wistful examination. Notion examination permits client to get a handle about what the survey are made with respect to explicit item or progressing matters as opposed to seeing profound considerations of the analyst. The content is examined for deciding opinion of the individual or gathering of people and arranged into constructive, pessimistic and unbiased. Twitter work one among the chief acknowledged stage to pass on the suppositions and perspectives on a chose item or theme. If individuals discover points significant or intriguing, at that point they may want to impart their insight about the subject. the subject may be an item or a help or a social message or the other article. Understanding this will assist us with choosing the kind of posts the client must increment such things for more client commitment through different ways. Investigation of web-based life realities like tweets encourages individuals to instigate an idea of a chose item or subject which settles on it simple to require choices subsequently. Assumption investigation additionally encourages individuals to adjust their mentality about wrong conviction on an item, administration or a subject. It causes individuals to choose which is best by investigating the remarks or tweets of a chose theme or an item. For e.g.: During races individuals experience the audits to frame it easy to make your psyche up or get an idea on whom to choose or choose the right applicant [3].

B. COMMUNITY DETECTION

An audit of network recognition in Twitter is introduced, uncovering philosophies with respect to the sort of network looked for, highlights, organize portrayal plot, network location/grouping calculations, and assessment measurements. Since the exhibition of strategies can shift radically per the database, results don't appear to be thought about [7].

A strategy to get geographic put together networks shaped with respect to Twitter at a situation of tropical storm Haiyan. The methodology speaks to informal community by weighted and non-coordinated chart where hubs indicate clients, which are associated by edges. The creators look for four changed diagrams and break down which produce the premier critical outcomes. Each diagram has its edges built with various traits: the essential chart utilizes normal notices among clients; the subsequent utilizes regular adherents; the 3rd is predicated on distributed URLs, and the 4th assumes basic words present inside the literary assemblage of texts. To implement network identification, an advanced adaptation of the FGM (Fast-avaricious Optimization of Modularity) calculation and the VDBSCAN calculation were utilized. Outputs were assessed by measurements that incorporate immaculateness, review and accuracy [7].

A system created for client division regarding commitment. A contextual analysis was led on Facebook. In any case, investigations might be summed up to other informal organizations, including Twitter. The philosophy comprises of three sections as mentioned below [7]:

(1) quantity of preferences a client provided on a chosen quantity of tweets.

(2) quantity of remarks a client has provided on a chosen quantity of tweets.

(3) Points resulted to a client indicating what amount the full remarks of the client on tweets are +ve and -ve.

The element (3) is figured utilizing a notion investigation calculation, though the others might be gotten after certain changes in information gathered legitimately from the system.

The exploration examines the matter of distinguishing networks in Twitter upheld clients' inclinations. to manage this issue, a weighted and non-coordinated diagram speaking to the system is fabricated, where edge loads are determined bolstered the closeness between clients, signified by hubs. The comparability between two clients u_i and u_j is registered utilizing the ensuing highlights: regular subjects found on literary group of tweets by LDA (Latent Dirichlet Allocation) model, normal URLs, basic # tags, quantity of basic companions and supporters, quantity of times u_i and u_j retweeted the indistinguishable individual, and subsequently the occasions u_i and u_j retweeted each other. At that point, the K-Means calculation is utilized to spot network structure [7].

The people group location issue in interpersonal organizations ought to be multifaceted, and it's imperative to require the consequent advances: comprehend which sorts of networks is additionally present inside the system, every one of which uncovers a feature of the network figure build up a system portrayal plan to respond to the inquiries expressed inside the past advance, and apply a network location calculation to disclose the network structure; look at changed networks that show up inside the winds up so as to uncover all the aspects present inside the network structure of the informal organization. The creators present a procedure that catches a few sorts of networks on Twitter, including those upheld social association, action, subjects of intrigue and communication. Right off the bat, the informal community is spoken to by a weighted and coordinated chart, where edges speak to the similitude between hubs, signified by clients. Also, the sides are registered with regards to a metric that utilizes three sorts of

data (every one of which equal to some aspect), that are: sum between tweets put together by a given client (networks bolstered movement); number of basic hashtags between two clients (networks upheld subjects of intrigue); how regularly two clients retweet and notice each other (connection based networks). At long last, the Order Statistics Local Optimization Method (OSLOM) calculation is implemented, and hence the nature of outcomes is assessed [7].

The examination of ongoing task displays the network recognition in Twitter leading as an incremental procedure comprising basically of 5 stages:

(1) information extraction: Data is removed legitimately from the system or acquired from another source, trailed by preprocessing steps.

(2) development of the system portrayal conspires: the improvement of a system portrayal plot is performed.

(3) network identification: A people group discovery calculation is applied.

(4) assessment: The distinguished gatherings are thoroughly assessed, physically and iteratively.

(5) examination of results.

The got groups are marked.

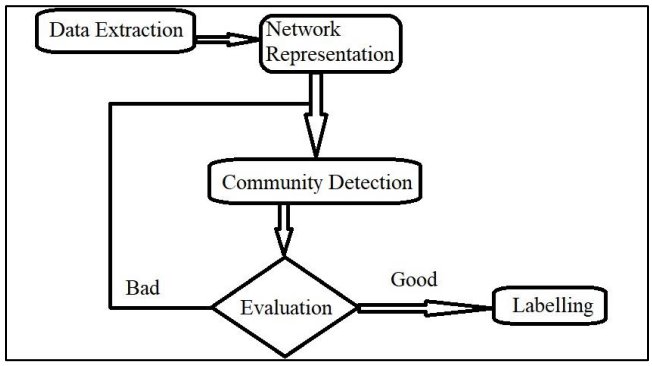

Fig 1. clusters labelled through community identification

The Fig. 1 delineates how network identification has been directed inside the setting of Twitter. it's imperative to see that the choice of system portrayal highlights and plans has been done with regards to what the specialists characterize as significant. Albeit a given arrangement of qualities, parameters, and calculations can deliver sensible outcomes for a provided data store. Advancing to limit such issue, our project proposes an advanced adjustment of present methodologies for network identification in Twitter [7].

The definition of a network fluctuates relying on the predetermined experiences. for example, inside the setting of Twitter, a few works investigate for networks of clients with comparative premiums, clients who collaborate every now and again, and clients who are geologically close to each other. In this manner, fixing the info chart for network identification is fundamentally guided into the objective application. Be that as it may, a large portion of those works are centered around identifying networks bolstered unequivocal connections between clients like devotee following connections, or notice/retweet connections. A frequently neglected wellspring of information is simply the substance of the tweets. In some utilization cases, these probably won't be significant. except for different cases, such as distinguishing networks of collaborating clients who share comparable suppositions towards specific points (hereinafter raised as conclusion-based connection networks), they will possibly improve network discovery. Network recognition generally yields heaps of networks, a considerable lot of which are exceptionally little ones and might be considered as "clamor" [6].

PROJECT WORK

Tools used: To perform various operations for this project we have used the following tools.

- **Jupyter Notebook:** open source web-application that enables us to execute big data analysis by using tools such as spark, python and r.
- **Gephi:** visualization software for different types of graphs. It has a lot of customizable plugins, metrics and manipulation tools.

Dataset: We have used live data from the twitter data stream for this project. To get this data from twitter we have applied for twitter developers account. After the access was granted, we created an app in the developer account called NDA_Project (App ID – 17493700). Once the app is created, we can get required keys which are used in the python tweepy library to connect to twitter API. To check the twitter API connectivity, we created a tweet from Jupyter notebook console.

For sentiment analysis we are getting latest ten thousand tweets which have corona virus as one of the words in the tweet. The data we get is raw and cleaning of data is required.

For Community detection we are getting the user account details screen name, username, number of followers, number of friends (following), number of tweets and account created timestamp. Also tweets of the user and tweets of his followers based on which we calculate the edge weight.

Libraries used: Python has wide range of libraries for this project implementation we have used the following libraries

- **Tweepy:** is a widely used python library which is used to access twitter API with python.
- **Pandas:** is a library built in python which is fast and powerful data analysis and manipulation tool.
- **Re:** regular expressions library is used for text manipulation in python.
- **Nltk:** is a natural language processing python library used for cleaning the data for sentiment analysis.
- **Collections:** provide containers for storing huge data in python.
- **Itertools:** is a part of collections module used to iterate the collections in python.

- **Networkx:** is python library used to study and create complex networks.
- **Matplotlib:** python library used for data visualization by creating graphs.
- **TextBlob: used** for determining the sentiment polarity and subjectivity of the words.
- Other libraries like **ssl, requests.exception** and **requests.packages.urllib3.exceptions** are for error handling when there is connectivity issue. For sentiment analysis we have used tweepy, pandas, re and nltk python libraries and for community detection we have used tweepy, networkx, matplotlib, csv and other python libraries like ssl, request.exception and requests.packages.urllib3.exceptions.

APPROACH FOLLOWED

A. SENTIMENT ANALYSIS

First, we download the required packages in anaconda prompt by using pip install library name and create a new workspace in Jupyter notebook and import the required libraries like tweepy, pandas, re and nltk. Then we open our app (NDA_Project) which we have created in twitter developers account and get the consumer key, consumer secret, access token secret and access token which we are going to use in the python workspace to connect to the twitter API. To check the connectivity, in the workspace by using tweepy's update_status method we can create a tweet and checks if its reflecting in the user timeline. Once the connectivity has been set up, we now proceed to get the twitter stream data and store in a list. In order to the required data we must pass the string (in our case I passed the string 'corona+virus') which will be used to get the tweets (tweepy Cursor method is used to get 10000 tweets were used to perform sentiment analysis). We have removed the retweets as we do not want any repeated tweets in our data. Then we defined a function clean_url which takes a tweet as parameter and cleans the tweets by removing "([^0-9A-Za-z \t])|(\w+:\/\/\S+)" these characters in the tweets and replacing with "". The cleaned tweets data will not have any URL's or any other special characters. The second stage of the cleanup process is to convert all the tweets into lowercase and split every tweet into words. This is performed by making use of in-built lower method and split method. Now we make itertools to flatten the list and create a bar graph by plotting the "most common words in the tweet". In this graph we can see that words which are most common are words like 'a', 'the', 'and' etc (these are words with no
meanings or stopwords). Then we download the stop words which are in nltk library and create a variable to store the stop words which are in English language. Iterate over the tweets again to remove the stop words in the tweet. By using itertools flatten the list and plot the graph again "most common words in the tweet – without stop words". As you can see that the bar graph which is now generated has corona and virus as the most frequently used words this because we search the tweets with "corona+virus". These are called collection words and the 3rd bar graph we visual the most common words in the tweets – without stopwords and collection words. This the final bar graph has most used words in the tweets and the words represented in the graph are meaning full which were most used in the tweets.

For calculating the sentiment of the tweet, we are using TextBlob python library which gives us the word subjectivity and polarity. We pass the all tweets which were generated after we pulled from for the search word "corona+virus" and after we have pass it to clean_url method to remove the URL's, symbols etc. Now we iterate textblob for each tweet in the list and create a pandas dataframe with columns polarity and tweet. After which we can go ahead and plot the graph for with number of tweets in y-axis and sentiment polarity ranging from -1 to 1 with a difference interval 0.25. we can observe that the there are way too many tweets with polarity equal to zero. For better visualization we have removed all the tweets with sentiment polarity equal to zero and plot the graph. Now we can clearly the sentiment of all the latest 10000 tweets in a graphical representation.

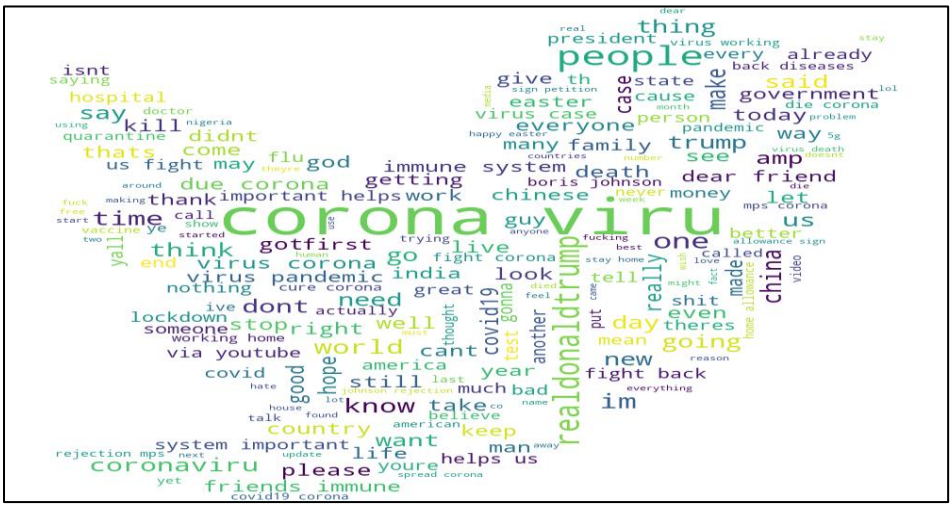

Fig 2. Word Cloud of tweets

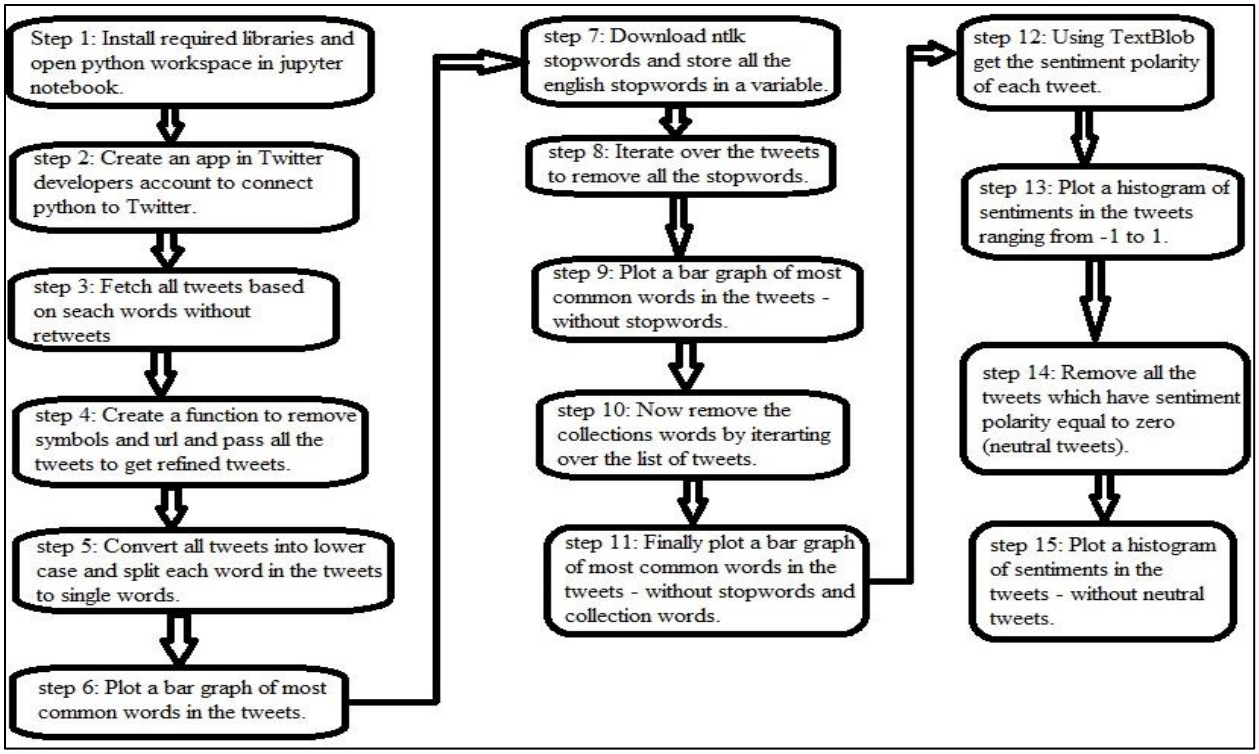

Fig 3. Flow chart for Sentiment Analysis

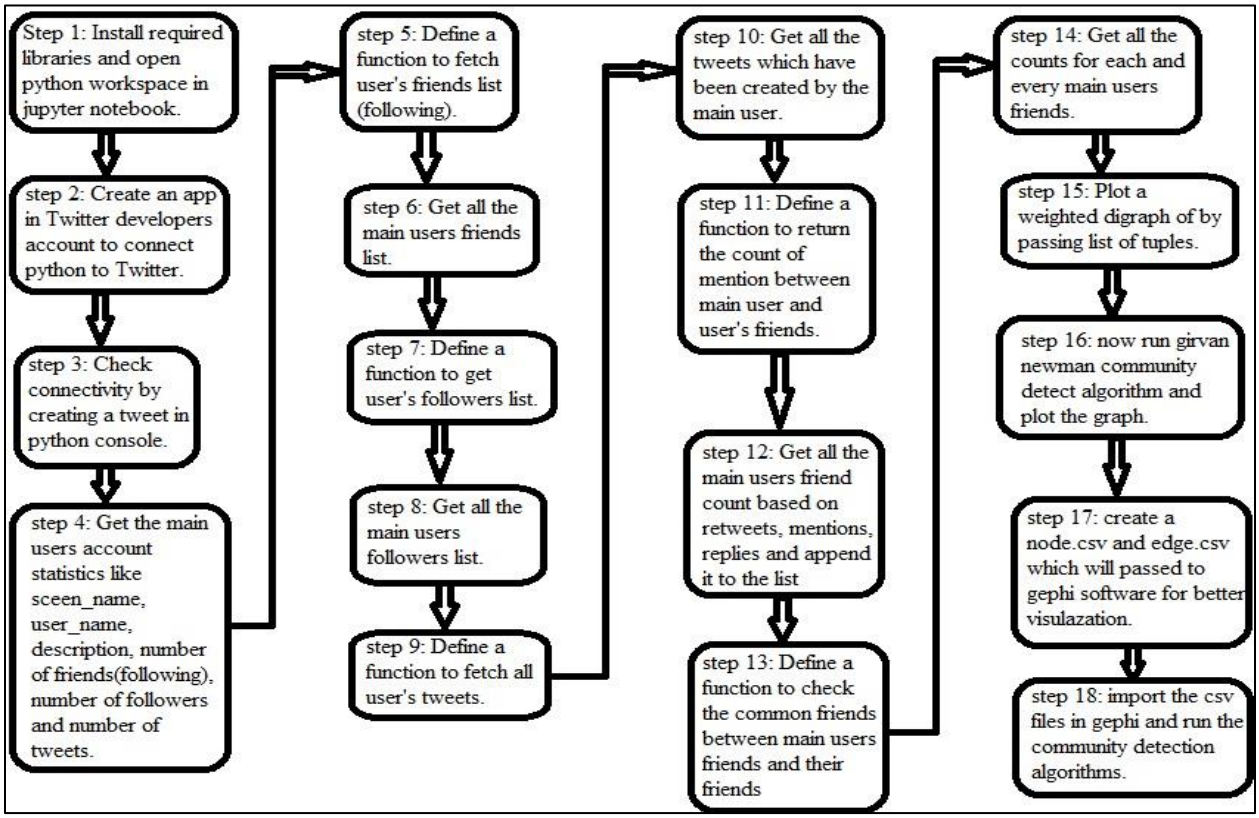

Fig 4. Flow chart for Community Detection

B. COMMUNITY DETECTION

like sentiment analysis First, we download the required packages in anaconda prompt by using pip install library name and create a new workspace in Jupyter notebook and import the required libraries like tweepy, networkx, matplotlib, csv and other python libraries like ssl, request.exception and requests.packages.urllib3.exceptions.. Then we open our app (NDA_Project) which we have created in twitter developers account and get the consumer key, consumer secret, access token secret and access token which we are going to use in the python workspace to connect to the twitter API. To check the connectivity, in the workspace by using tweepy's update_status method we can create a tweet and checks if its reflecting in the user timeline. Once the connectivity has been set up, we now proceed to get the main user statistics like screen name, username, description, number of friends(following), number of followers and number of tweets

Then define the following function

- get_friends(username) – this function returns the user's friends (following) in the form of list by taking the username as input parameter.
- get_followers(username) – this function returns the user's followers in the form of list by taking the username as input parameter.
- get_user_tweets(username) – this function returns the all the tweets which has written from his timeline.
- countPerFollower(username, user_tweets, handle_name) – this method takes the above parameters and check if the handle_name has been mentioned by the user in retweets or replies and returns a tuple containing username, handle_name and count which will be the input to generate weighted DiGraph.
- get_common_friends(main_user_list, friend_friend_list) – this method is takes 2 lists of follower and returns the common friends in the list as a list.
- get_nodes_edges(start_range, end_range) – this method calls other methods like get_friends_list, get_common_friends, get_user_tweets and countsPerFollower. This is the main method which calculates the nodes, edges and the weight.

First, we get the main user's friends and store them in a list. Followed by the main user's tweets. Based on which we calculate the count and add them to the list. Then by using the get_nodes_edges we pass the friends list of the main user in batches. Iterate the batch list and get the friend list of that user using get_friends method and check if the main user has common friend by using the get_common_friends method and if the returned list not empty we get that friends tweets and calculate the count of mentions in the user timeline. The returned tuple value will be appended to the list which will be used to generate the directed graph with weights. After which we concert the directed graph to undirected graph and run the inbuilt method girvan_newman which will return the communities by removing the links.

To show better visualization we export the data which we got into two csv files called node.csv and edge.csv which will be used as input for Gephi. In Gephi we import these tables and change all the weight value of zero to one. Now we set the node diameter based on the number of out-links and the edge weight based on the weight we calculated in python code. Now we run the modularity to detect the various communities in the network.

ANALYSIS & EXPERIMENTAL RESULTS

In our first experimental results we can see that we have connected to twitter API and created a tweet form our python code. To perform this, we have used update_status method from tweepy library after we enter the 4 keys to connect our python code to the app, we developed in developers account.

Fig 5. Tweet from python using tweepy library

We define the search words "corona+virus" and remove retweets in order to have unique tweets. We only fetch the content in the tweet using the text attribute.

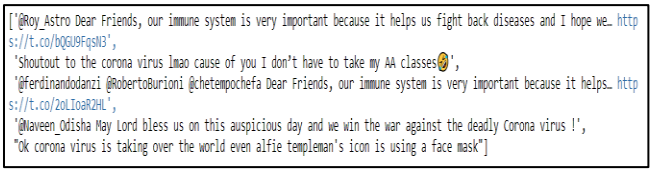

Fig 6. Tweets based on search word

Next we create a method called clean_url all pass all the tweets we fetched to remove symbols and URL in the tweet using the regular expressions library re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", tweet) and replace it with empty space.

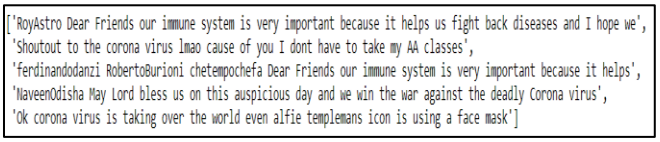

Fig 7. Cleaned tweets after removing symbols and URL's

Then we further clean the tweet by converting all the words into lowercase using the inbuilt lower method and split the sentences in tweets to words by using split method.

```
Out[4]: [['royastro',
          'dear',
          'friends',
          'our',
          'immune',
          'system',
          'is',
          'very',
          'important',
          'because',
          'it',
          'helps',
          'us',
          'fight',
          'back',
          'diseases',
          'and',
          'i',
          'hope',
          'we'],
         ['shoutout',
          'to',
```

Fig 8. Tweets after converting separate words and lowercase

Now we plot the horizontal bar graph of the fifteen most common tweets for the search words. We can see that most of the words in the tweet are commonly used words in English language like 'a', 'the', 'and' etc.

In order to remove these stopwords we use the help of nltk (natural language processing library) in python to remove these words from the list of words in all the tweets we downloaded. Then we plot a horizontal bar graph again to show the fifteen most common words in the tweets - without stopwords.

```
Out[7]: [['royastro',
          'dear',
          'friends',
          'immune',
          'system',
          'important',
          'helps',
          'us',
          'fight',
          'back',
          'diseases',
          'hope'],
```

Fig 9. Tweets after removing the stopwords

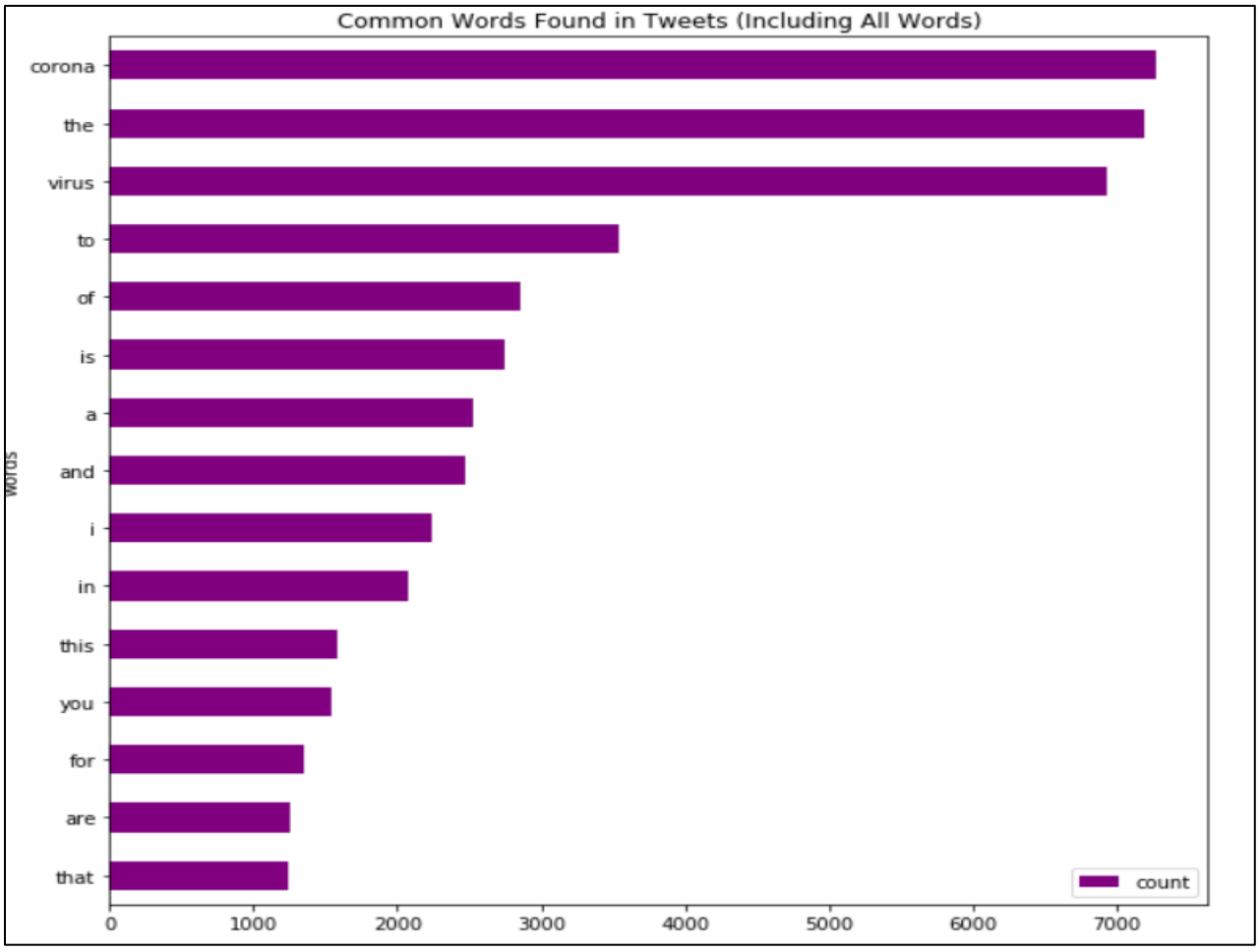

Fig 10. Common Words Found in Tweets (with stopwords)

We can see that most common words are corona and virus which are search words used to get the data. These are called collection words. In the next horizontal bar graph, we plot is fifteen most common words used in the tweets – without stopwords or collection words.

In order to calculate the sentiment of tweets we make use of TextBlob python library. Then get the polarity of each tweet and store them in a panda's data frame.

Now plot a histogram for the data frame. The graph produced is with number of tweets on x-axis and sentiment polarity on y-axis. The graph produced is "sentiment polarity of the tweets"

For better visibility lets remove the neutral tweets and plot the histogram which will give us "sentiment polarity of tweets – without neutral tweet.

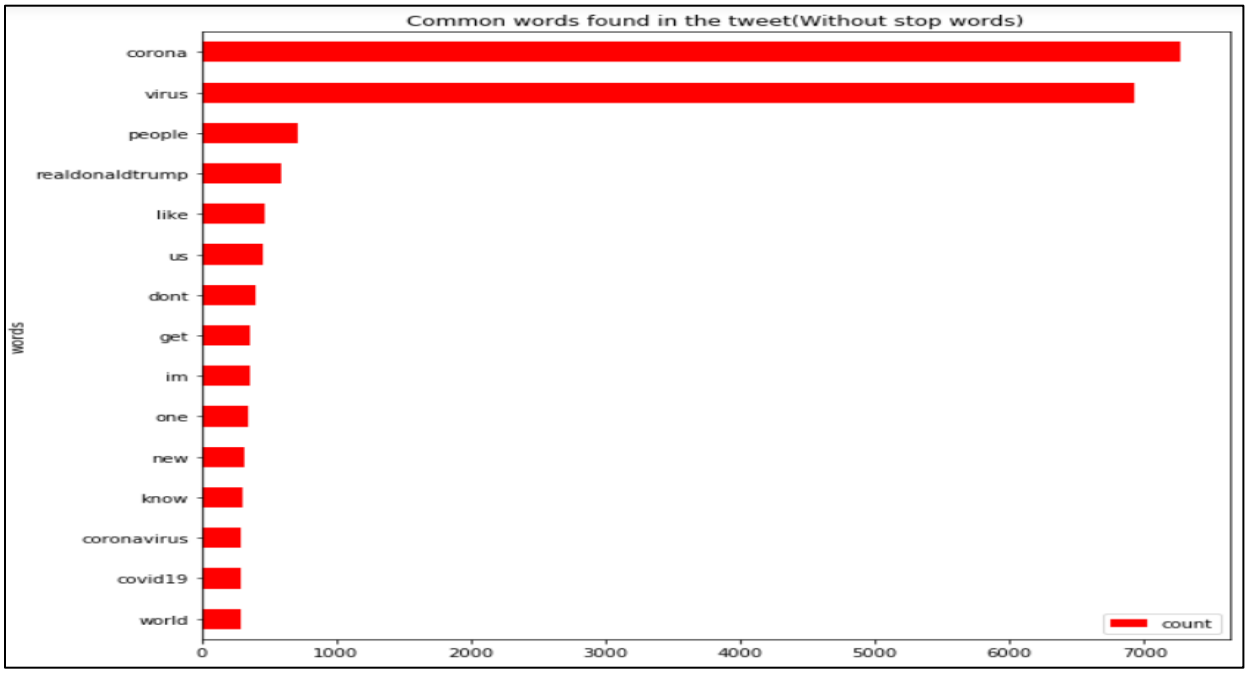

Fig 11. Common words found in tweets (without stopwords)

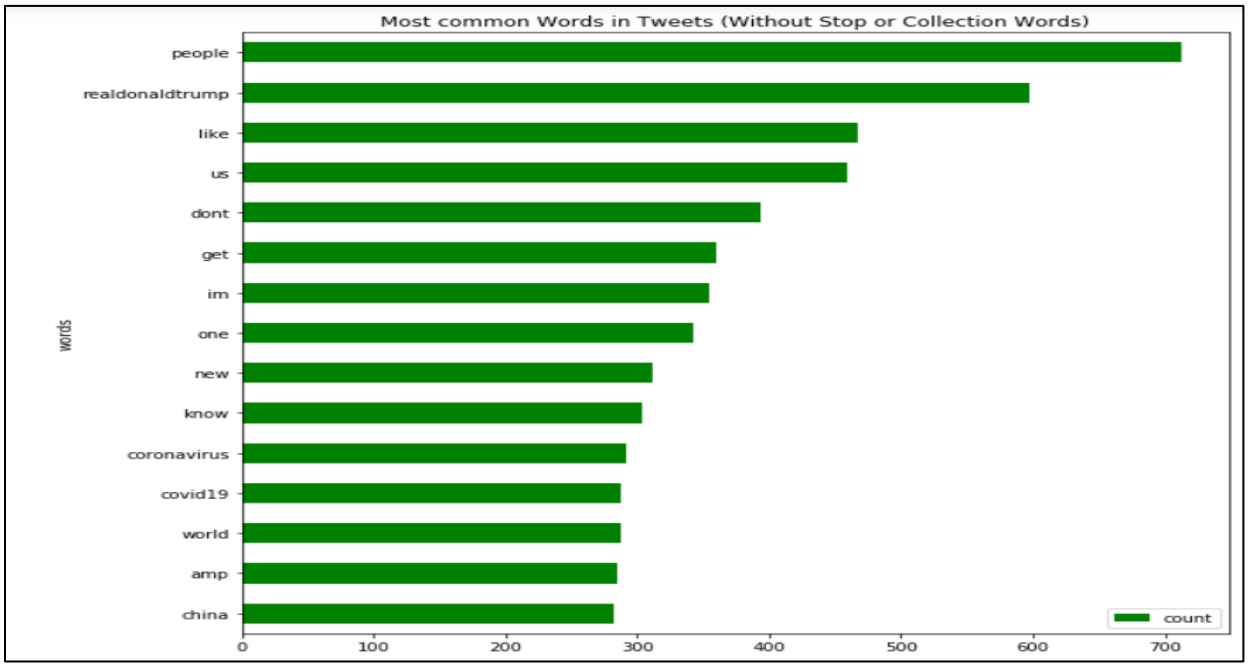

Fig 12. Common words found in tweets (without stopwords and collection words)

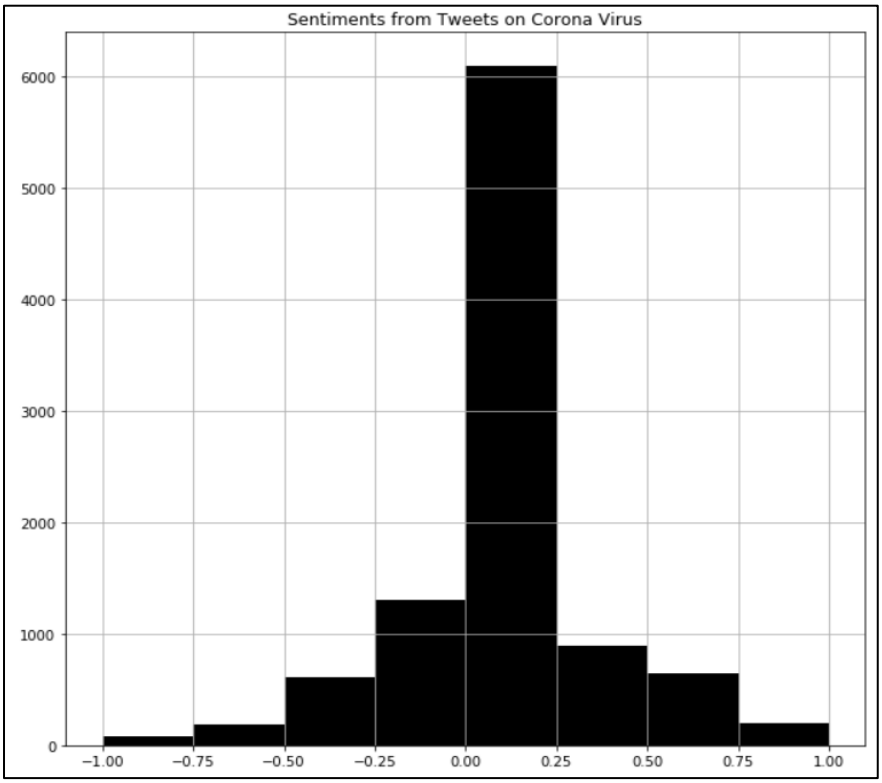

Fig 13. Sentiment Polarity of tweets

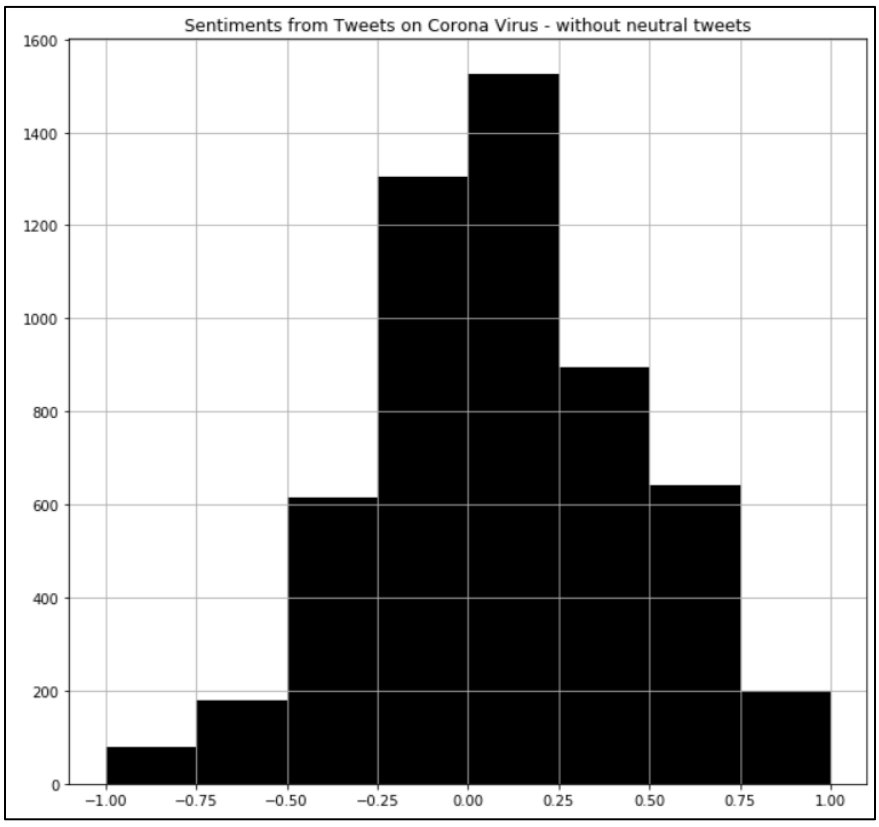

Fig 14. Sentiment Polarity of tweets without neutral tweets

Coming to Community detection lets first assign a main user for which we detect the community for. Then with the help of tweepy we can get the user details like username, screen name, number of followers, number of friends (following) and number of tweets in user timeline.

```
name: Aakash Bakhle
screen_name: aakashb_95
description: Everyday photographer, observer.
statuses_count: 564
friends_count: 189
followers_count: 81
account_created_date: 2013-03-02 11:15:42
```

Fig 15. Main user statistics

Now we define a function which returns a list of user's friends when we pass username.

```
Out[16]: ['YoBlackPepper',
          'dailyherapheri',
          'bhaleraosarang',
          'AkkiMB5',
          'Sand5066',
          'naval',
          'salty__bae',
          'dansiddiqui',
          'MajorNeel',
          'NiTiSHmurthy',
          'AmbujaNaik',
          'labnol',
          'GappistanRadio',
          'JackMa',
          'deepigoyal',
          'Sassy_Soul_',
          'PTI_News',
          'ANI',
          'delhichatter',
```

Fig 16. Users Friends list

Create another function to return the list of user's followers list when we pass a username.

```
Out[18]: ['AkshayIndalkar6',
          'AkkiMB5',
          'Sand5066',
          'salty__bae',
          'FFH_HQ',
          'notdeepblue',
          'R4G3xHOPE',
          'alisha_janrao',
          'DaithankarNinad',
          'tej95',
          'BakhleShrirang',
          'Amit28824553',
          'Manc43ver',
```

Fig 17. Users Followers list

Create a function to return the list of user tweets in a list when we pass username to it

```
['@nocontextunited @letmehandlek 😂😂😂😂',
 "3 years since #bedsforawayfans. Perhaps the most beautiful gesture I've witnessed live.\n@BVB https://t.co/M
 '@RandomCricketP1 @mandar_94 @letmehandlek @thatsharmaboy',
 'RT @dailyherapheri: Indian parents to their kids during summer vacations https://t.co/5b9Yq7e14M',
```

Fig 18. Users tweets from timeline

Define a function to return tuple of username, handle name and count when we pass username, user tweets and handle name. This is then stored in a list which is passed as input to digraph.

```
Out[22]: [('aakashb_95', 'YoBlackPepper', 0),
          ('aakashb_95', 'dailyherapheri', 1),
          ('aakashb_95', 'bhaleraosarang', 0),
          ('aakashb_95', 'AkkiMB5', 0),
          ('aakashb_95', 'Sand5066', 0),
          ('aakashb_95', 'naval', 0),
          ('aakashb_95', 'salty__bae', 0),
          ('aakashb_95', 'dansiddiqui', 0),
          ('aakashb_95', 'MajorNeel', 0),
          ('aakashb_95', 'NiTiSHmurthy', 0),
          ('aakashb_95', 'AmbujaNaik', 0),
          ('aakashb_95', 'labnol', 0),
          ('aakashb_95', 'GappistanRadio', 2),
          ('aakashb_95', 'JackMa', 0),
```

Fig 19. Main users edge count between friends

Define a function to check the common friends we pass the list of main users' friends and list of main users' friends friend list. The output of this function will be a list of common friends.

Now define a main function to call the above function recursively for all the users so that we get all edge weight in the form of tuple which will later be appended to the main list that will be used for generating the directed graph.

```
('richardbranson', 'BillGates', 0),
('richardbranson', 'LeoDiCaprio', 0),
('richardbranson', 'TheEconomist', 0),
('richardbranson', 'elonmusk', 0),
('richardbranson', 'NASA', 1),
('richardbranson', 'StephenAtHome', 0),
('elonmusk', 'NASA', 2),
('elonmusk', 'Tesla', 27),
('elonmusk', 'SpaceX', 6)]
```

Fig 20. Common friends edge count between users

Using network, we can create the directed graph by passing the list of tuples which has source node, destination node and count which acts as weight. By using the matplotlib python library we plot and show the graph.

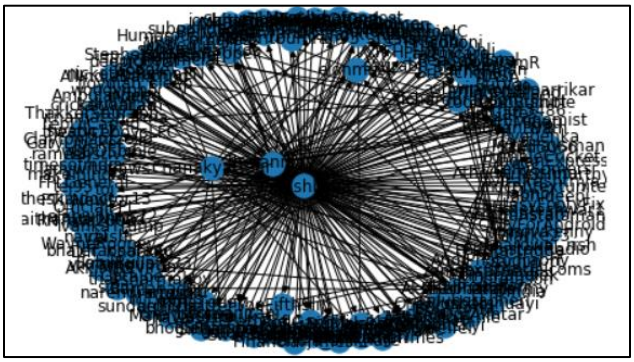

Fig 21. Directed Graph of users

In order to perform girvan newman inbuilt algorithm which only accepts an undirected graph we convert the directed graph to undirected graph using the method nx.DiGraph.to_Undirected. Which is again plotted using matplotlib.

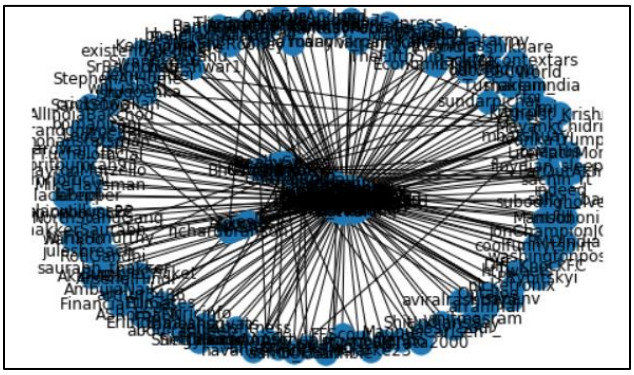

Fig 22. Undirected Graph of users

After running the girvan_newman community detection algorithm we get a list of lists as result which has of different nodes in different communities. Girvan_newman algorithm works by removing the edge based on the high edge centrality. The edges are kept on removed until we get different communities in the network.

```
    'warikoo',
    'washingtonpost',
    'will_iamR',
    'woodyinho',
    'year_progress',
    'zenlabsfitness'],
  ['YoBlackPepper'],
  ['bhaleraosarang']]
```

Fig 23. Communities detected using Girvan Newman method

To create a node csv file, we make use of csv python library to write the content of the main users' friends list into it. There will be two columns id and name. we are only passing the id in the csv file and later we are manually editing the csv file to add the names.

	A	B
1	id	name
2	dailyherapheri	dailyherapheri
3	bhaleraosarang	bhaleraosarang
4	AkkiMB5	AkkiMB5
5	Sand5066	Sand5066
6	naval	naval

Fig 24. Node.csv table for Gephi input

Similarly, to create the edge csv file we are also making use of csv python library to write the content of the edge list of tuples which has source node, destination node and weight into three different columns – source node, destination node and weight in the csv file. Now we manually edit the csv file to add type column between destination node and weight and fill values of 'directed' in them.

	A	B	C	D
1	source	target	type	weight
2	aakashb_9	dailyherap	directed	1
3	aakashb_9	bhaleraosa	directed	0
4	aakashb_9	AkkiMB5	directed	0
5	aakashb_9	Sand5066	directed	0

Fig 25. Edge.csv table for Gephi input

In Gephi software we import both the csv files. We first change the value of zero in weight of edge table to one. In the node tab set the node width based on the out degree and weight of the edge is set by weight we passed in the csv file. Then we run the modularity which is required to detect communities in the network.

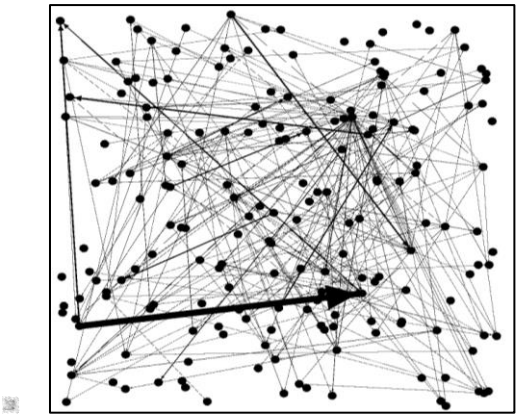

Fig 26. Graph structure after Gephi import

The above image is when we initially import both the node.csv and edge.csv into a single Gephi workspace in the overview window. As the type is directed we can see the arrows pointing towards nodes.

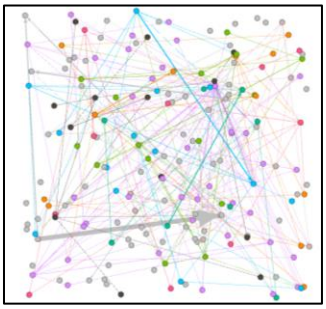

Fig 27. Intermediate graph in Gephi

The Graph on the left is the intermediate result obtained after we run the modularity and set community detection appearance window.

The graph on below is the final graph which we obtain after we have set all the parameters and run the force atlas 2 algorithm with no overlapping nodes and we can observe that there are a total of 8 different communities in the given network. The node with maximum out degree is the main user. Hence, we have successfully completed community detection for a given twitter user.

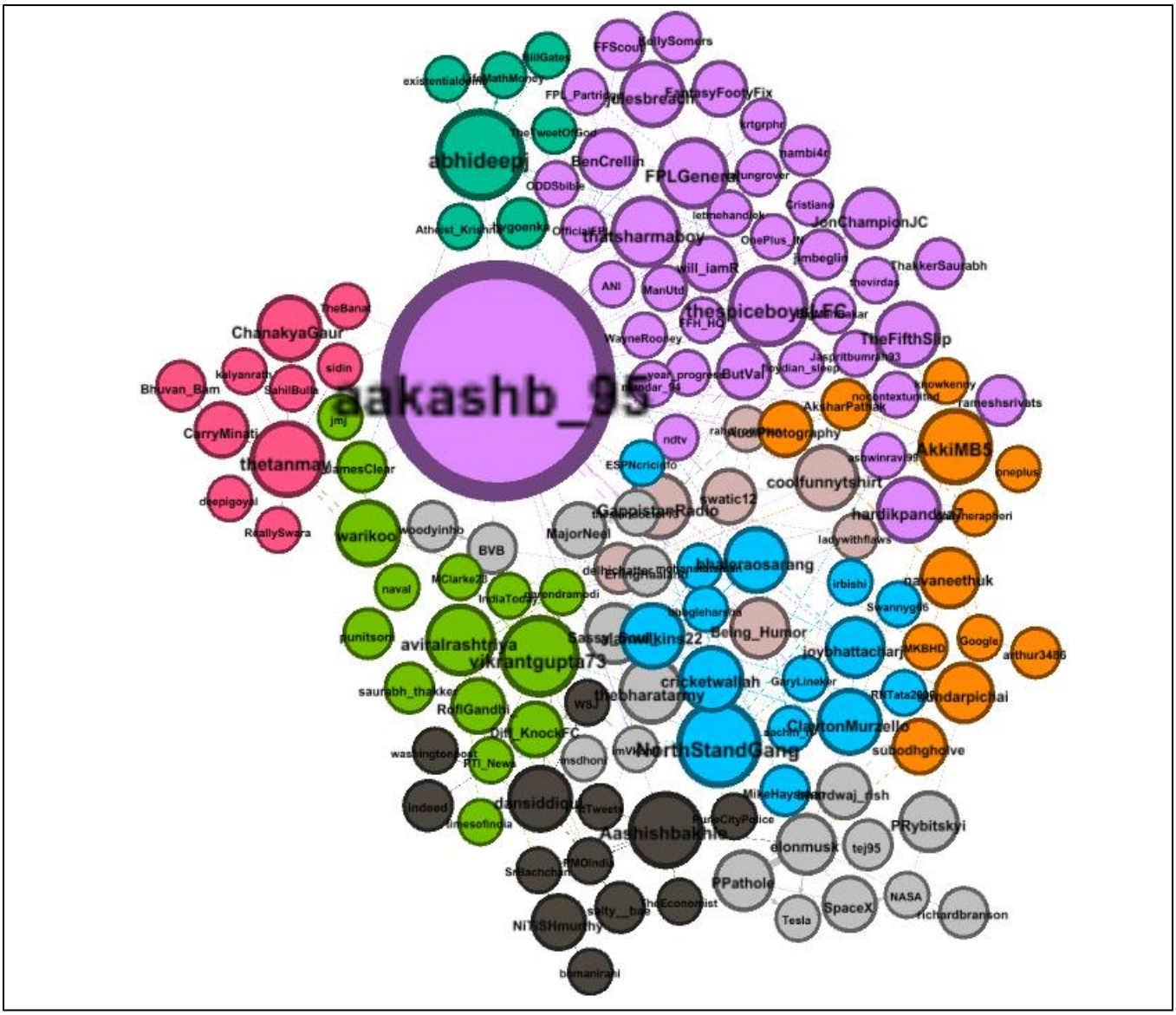

Fig 28. Community Detection in Gephi

ANALYSIS

The data set used for sentiment analysis in ten thousand tweets. Then this data is cleaned by removing the URL's, symbols and emojis. The we further clean the data by converting into lowercase and splitting each tweet into words. After the data is preprocessed, we go ahead creating our first horizontal bar graph obtained for sentiment analysis shows the most common words found in the tweet. Most of words are meaning less words or common words used in English language. In the second horizontal bar graph we overcome this with the help of nltk natural language processing library. This library has stopwords which contain words like 'a', 'an', 'the', 'etc'. we are plotting the graph by removing these words from the tweets. We can observe that the graph has corona and virus as topmost common words this is because we fetched tweets based on the search words "corona+virus". These are called collections words. In the next horizontal bar graph, we are going to plot the most common words found in tweets – without stopwords or collection words. After which we can proceed plotting a word cloud of five hundred words in the image with the help of numpy array and PIL python libraries. The word cloud is great way to visualize data and provides an instant understanding to the user on what the data is about.

After data preprocessing, we can now start getting the sentiment polarity of the tweets using the TextBlob python library and store the result in panda's data frame with two columns 'polarity' and 'tweet'. Now we pass this data frame to plot a histogram of tweet sentiments with number if tweets in y-axis and tweet polarity ranging from -1 to 1 which difference interval of 0.25. we can observe that there the histogram we go is not very clear as there a lot of tweets ranging between 0 and 0.25. In order to improve the view lets remove the neutral tweets and plot the histogram again. Now we can observe better results and by analyzing the histogram we can see that there are a greater number of positive tweet than negative tweets for the search words "corona+virus".

For Community detection we first connected to twitter API and assign a main user for which we perform community detection. For this user we get all the statistics like screen name, username, followers count, following count, number of user tweets in the timeline and account created date. After we define several methods to get the count. The count then passed to a directed graph for plotting. We can see that the network is very interconnected that is some users have many common connections where some have only few connections. Connection frequency between normal user and normal user is very low whereas there a lot of connection between normal user and verifies used (google, tesla, spacex). For creating a network graph, it took a lot of time to get the data and in some cases data retrieval failed due to connection timeout. Hence had to process in batches.

After we complete community detection in Gephi we see a total of eight different communities by running the modularity and setting the node diameter based on out links and edge width by setting the weight we passed.

CODE EXPLAINATION

Import required packages and connect to twitter API

```
import os
import tweepy as tw
import pandas as pd
import math

import re
import nltk
from nltk.corpus import stopwords
import networkx as nx
from networkx.algorithms import community
import matplotlib.pyplot as plt

import collections
import itertools
from textblob import TextBlob

from socket import socket
import ssl
from requests.exceptions import Timeout, ConnectionError
from requests.packages.urllib3.exceptions import ReadTimeoutError, ProtocolError

import csv

import warnings
warnings.filterwarnings("ignore")


consumer_key='dUBhMzdBUyES8b5aOj5FZ1PG8'
consumer_secret='emsWamOq4SVNVf9b5cLOqUncen10slmXPt036vOh3Rw9IRRwKJ'
access_token='1230514693674094593-AtxhZ7C3n9QlI0ZevlBPCds7lPsjUK'
access_token_secret='rkpIruukE1xsT2vr4dE08q0HDECFIMOTXkrNY281qsWBO'

auth=tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api=tw.API(auth,wait_on_rate_limit=True,timeout=600)
```

Define search words for tweets extraction

```
search_words='corona+virus'
new_search=search_words + " -filter:retweets"
tweets=tw.Cursor(api.search,q=new_search, lang="en").items(10000)
all_tweets= [tweet.text for tweet in tweets]
all_tweets[:5]
```

Define function to clean symbols and URL's and call it

```
def clean_url(txt):
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt).split())

all_tweets_no_urls = [clean_url(tweet) for tweet in all_tweets]
all_tweets_no_urls[:5]
```

Preprocess tweets by converting into lowercase and splitting them

```
words_in_tweet=[tweet.lower().split() for tweet in all_tweets_no_urls]
words_in_tweet[:5]
```

Flatten the lest and plot "common words in tweets" bar graph

```
tweet_words= list(itertools.chain(*words_in_tweet))
tweet_words_count= collections.Counter(tweet_words)


clean_tweets_with_stopwords= pd.DataFrame(tweet_words_count.most_common(15), columns=['words','count'])

fig, ax=plt.subplots(figsize=(10,10))

clean_tweets_with_stopwords.sort_values(by='count').plot.barh(x='words', y='count', ax=ax, color="purple")

ax.set_title("Common Words Found in Tweets (Including All Words)")

plt.show()
```

Download stopwords and remove them from existing tweets

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sand5\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
```

```
stop_words= set(stopwords.words('english'))

tweets_new =[[word for word in tweet_words if not word in stop_words] for tweet_words in words_in_tweet]
tweets_new[:5]
```

Plot "common words in tweets – without stopwords"

```
all_words= list(itertools.chain(*tweets_new))
count_words= collections.Counter(all_words)
#count_words.most_common(20)

clean_tweets= pd.DataFrame(count_words.most_common(15),columns=['words','count'])
clean_tweets

fig, ax= plt.subplots(figsize=(10,10))

clean_tweets.sort_values(by='count').plot.barh(x='words', y='count', ax=ax, color="red")

ax.set_title("Common words found in the tweet(Without stop words)")

plt.show()
```

Remove collection words and Plot the final bar graph

```
collection_words= ['corona', 'virus']

tweets_no_collection_words= [[tweet_no_collection_word for tweet_no_collection_word in word
                              if not tweet_no_collection_word in collection_words] for word in tweets_new]

all_words_nc= list(itertools.chain(*tweets_no_collection_words))
count_words_nc= collections.Counter(all_words_nc)

#count_words_nc.most_common(15)

clean_tweets_nc=pd.DataFrame(count_words_nc.most_common(15), columns=['words','count'])

fig, ax=plt.subplots(figsize=(10,10))

ax.set_title("Most common Words in Tweets (Without Stop or Collection Words)")

clean_tweets_nc.sort_values(by='count').plot.barh(x='words', y='count', ax=ax, color="green")
```

Code for creating word cloud

```
import numpy as np
from PIL import Image
from wordcloud import WordCloud, STOPWORDS


def create_wordcloud(text):

    mask = np.array(Image.open("twitter_mask.png"))
    stopwords = set(STOPWORDS)

    wc = WordCloud(background_color="white", max_words=200, mask=mask, stopwords=stopwords)
    wc.generate(text)

    wc.to_file( "tm.png")

str2=" ".join(all_words)

create_wordcloud(str2)
```

Using TextBlob get the sentiments and store them in data frame

```
sentiment_objects = [TextBlob(tweet) for tweet in all_tweets_no_urls]

sentiment_values = [[tweet.sentiment.polarity, str(tweet)] for tweet in sentiment_objects]

sentiment_df = pd.DataFrame(sentiment_values, columns=["polarity", "tweet"])

sentiment_df.head()
```

Plot the histogram of tweets sentiments

```
fig, ax = plt.subplots(figsize=(10, 10))

sentiment_df.hist(bins=[-1, -0.75, -0.5, -0.25, 0.0, 0.25, 0.5, 0.75, 1], ax=ax, color="black")

plt.title("Sentiments from Tweets on Corona Virus")
plt.show()
```

Remove neutral tweets and plot the histogram again

```
sentiment_df=sentiment_df[sentiment_df.polarity!= 0]
#sentiment_df.head()

fig, ax = plt.subplots(figsize=(10, 10))

sentiment_df.hist(bins=[-1, -0.75, -0.5, -0.25, 0.0, 0.25, 0.5, 0.75, 1], ax=ax, color="black")

plt.title("Sentiments from Tweets on Corona Virus - without neutral tweets")
plt.show()
```

Get user data for Community Detection

```
user_name="aakashb_95"

item= api.get_user(user_name)
print("name: " + item.name)
print("screen_name: " + item.screen_name)
print("description: " + item.description)
print("statuses_count: " + str(item.statuses_count))
print("friends_count: " + str(item.friends_count))
print("followers_count: " + str(item.followers_count))
print("account_created_date: "+ str(item.created_at))

user_tweet_count=item.statuses_count
```

Define function to get following list

```
def get_friends(usr_name):
    friends = []
    for user in tw.Cursor(api.friends, screen_name=usr_name).items():
        friends.append(user.screen_name)
    return friends
```

Define function to get followers list

```
def get_followers_list(usr_name):
    followers = []
    for user in tw.Cursor(api.followers, screen_name=usr_name).items():
        followers.append(user.screen_name)
    return followers
```

Define function to get user tweets

```
def get_user_tweets(usr_name):
    user_tweets= []
    for tweet in api.user_timeline(id=usr_name,count=200):
    #for tweet in tw.Cursor(api.user_timeline,id=usr_name).items():
        user_tweets.append(tweet.text)
    return user_tweets
```

Call all the above functions and store them in variables

```
#main_user_friends_list= []
main_user_friends_list=get_friends(user_name)
#main_user_friends_list
main_user_followers_list= get_followers_list(user_name)
#main_user_followers_list
main_user_tweets= get_user_tweets(user_name)
#main_user_tweets
```

Define function return common friends

```
def get_common_friends(main_user_list, friend_friends_list):
    #print("length from common_friends main list:"+len(main_user_list))
    list1_as_set = set(main_user_list)
    intersection = list1_as_set.intersection(friend_friends_list)
    intersection_as_list = list(intersection)
    #print("Common friends: "+intersection_as_list)
    return intersection_as_list
```

Define function to get edge count

```
def countsPerFollower(usr_name,usr_tweets, handle_name):
#     temp = []
    count = 0
    search_str = '@'+handle_name
    for tweet in usr_tweets:
        count += tweet.count(search_str)
#         temp.append((item.screen_name, handle_name, tweet.count(search_str)))

    return (usr_name, handle_name, count)
```

Get the mains users edges and store them in a list

```
l = [countsPerFollower(user_name,main_user_tweets,i) for i in main_user_friends_list]
l
```

Main function which calls all other functions recursively

```
def get_nodes_edges(start_range,end_range):
    chunk_list=main_user_friends_list[start_range:end_range]
    print(chunk_list)
    for i in range(len(chunk_list)):
        try:
            friend_friends= get_friends(chunk_list[i])
            print('Source node: '+chunk_list[i])
            print("**********************************")
            common_friends=get_common_friends(main_user_friends_list,friend_friends)
            #print(common_friends)
            if len(common_friends)!=0:
                friend_tweets= get_user_tweets(chunk_list[i])
                for j in range(len(common_friends)):
                    print('Destination node: '+common_friends[j])
                    friend_count= countsPerFollower(chunk_list[i],friend_tweets,common_friends[j])
                    #print(friend_count)
                    l.append(friend_count)
        except (Timeout, ssl.SSLError, ConnectionError, ReadTimeoutError, ProtocolError) as exc:
            time.sleep(10)
            print("error")
```

Calling the main function

```
# run in batches of 10 to avoid connection timeout
get_nodes_edges(131,140)
len(l)
l
```

Writing data into csv and later editing as per Gephi format

```
with open('edge.csv','w',newline='') as output:
    csv_out=csv.writer(output)
    csv_out.writerow(['source','target','weight'])
    for row in l:
        csv_out.writerow(row)
```

```
with open('node.csv','w',newline='') as output:
    csv_out=csv.writer(output)
    csv_out.writerow(['id','name'])
    for row in main_user_friends_list:
        csv_out.writerow(row)
```

Create a direct graph by passing list of tuples which has weight

```
G= nx.DiGraph()
G.add_weighted_edges_from(l)
pos = nx.spring_layout(G,k=5/math.sqrt(G.order()),iterations=20)
nx.draw(G,with_labels=True)
plt.draw()
plt.show()
```

Convert directed to undirected graph for community detection

```
G1=nx.DiGraph.to_undirected(G)
nx.draw(G1,with_labels=True)
plt.draw()
plt.show()
```

Calling Girvan Newman method to detect communities.

```
communities_generator = community.girvan_newman(G1)
top_level_communities = next(communities_generator)
next_level_communities = next(communities_generator)
sorted(map(sorted, next_level_communities))
```

CONCLUSION

In conclusion this paper talked about various related works on the sentiment analysis and community detection project we developed in the literature survey section, followed by tools, python libraries and datasets used in the project requirements section. Then we discussed the approach we followed to complete this project followed by the experimental results and various visual representation of data like word cloud, horizontal bar graphs, histograms, directed graph, undirected graph and finally the community detection graph which was generated in Gephi visualization software. After which we have also conducted an analysis on the results obtained. In the last section of the paper we have explained the code which we have developed to implement the project.

As a part of this project we have successfully implemented sentiment analysis and community detection for a single twitter user. The future works for this project are extending to Multi Class sentiment analysis that is seven different types sentiment classes (‘sad’, ‘Happy’, ‘neutral’, ‘fun’, ‘anger’, ‘hate’. ‘love’) and for community detection go deeper into several levels not just fetching the communities by 1 step neighborhood that is extending beyond friends or friends which will give us more accurate results.

REFERENCES

[1] C. R. Nirmala, G. M. Roopa and K. R. N. Kumar, "Twitter data analysis for unemployment
crisis," 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Davangere, 2015, pp. 420-423.
[2] F. Nausheen and S. H. Begum, "Sentiment analysis to predict election results using Python,"
2018 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, 2018,
pp. 1259-1262.
[3] V. Prakruthi, D. Sindhu and D. S. Anupama Kumar, "Real Time Sentiment Analysis of Twitter
Posts," 2018 3rd International Conference on Computational Systems and Information
Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 2018, pp. 29-34.
[4] Z. Jianqiang and G. Xiaolin, "Comparison Research on Text Pre-processing Methods on
Twitter Sentiment Analysis," in IEEE Access, vol. 5, pp. 2870-2879, 2017.
[5] M. Bouazizi and T. Ohtsuki, "A Pattern-Based Approach for Multi-Class Sentiment Analysis in
Twitter," in IEEE Access, vol. 5, pp. 20617-20639, 2017.
[6] A. J. Lam and C. Cheng, "Utilizing Tweet Content for the Detection of Sentiment-Based
Interaction Communities on Twitter," 2018 IEEE 5th International Conference on Data Science
and Advanced Analytics (DSAA), Turin, Italy, 2018, pp. 682-691.
[7] Wendel Silva, Ádamo Santana, Fábio Lobato, and Márcia Pinheiro. 2017. A methodology for
community detection in Twitter. In Proceedings of the International Conference on Web

Intelligence (WI '17). Association for Computing Machinery, New York, NY, USA, 1006–1009.
[8] M. van Meeteren, A. Poorthuis and E. Dugundji, "Mapping communities in large virtual
social networks: Using Twitter data to find the Indie Mac community," 2010 IEEE International
Workshop on: Business Applications of Social Network Analysis (BASNA), Bangalore, 2010, pp.
1-8.