Empirical Election Community Analysis and Prediction using Social Media Demographics

Submitted in partial fulfilment for the award of the degree

of

Bachelor of Technology

in

Computer Science and Engineering

by

Faraz Ahmad

School of Computer Science and Engineering

October, 2018

Abstract

The rapid growth of social media has given users a stage to voice their assessments and opinions. Organizations or entities need to recognize the polarity of these assessments with the end goal to understand user opinion and optimise the decision making process. This idea can be used extensively in the field of legislature where political parties need public opinion in order to increase their popularity and campaigning. Sentiment Analysis through social media has been seen as a successful apparatus to screen client inclinations and tendency. The following system uses Naïve Bayes, a supervised learning algorithm to prepare an informational index and analyse the different political entities in India's general elections of 2019. The following system aims at analysing the popularity and opinions on twitter without compromising on features and contextual relevance of textual data. The project uses K-means clustering in order to perform predictions based on popularity, sentiments and overall comprehension.

Keywords- text mining, supervised learning, Naïve Bayes, K-means, clustering

1. Introduction

Sentiment Analysis is process of collecting and analysing data based upon the person feelings, reviews and thoughts. Sentimental analysis often called as opinion mining as it mines the important feature from people opinions. Sentimental Analysis is done by using various machine learning techniques, statistical models and Natural Language Processing (NLP) for the extraction of feature from a large data. Sentiment Analysis can be done at document, phrase and sentence level. In document level, summary of the entire document is taken first and then it is analyse whether the sentiment is positive, negative or neutral. In phrase level, analysis of phrases in a sentence is taken in account to check the polarity. In Sentence level, each sentence is classified in a particular class to provide the sentiment. Sentimental Analysis has various applications. It is used to generate opinions for people of social media by analysing their feelings or thoughts which they provide in form of text. Sentiment Analysis is domain centred, i.e. results of one domain cannot be applied to other domain. Sentimental Analysis is used in many real life scenarios, to get reviews about any product or movies, to get the financial report of any company, for predictions or marketing. Twitter is a micro blogging platform where anyone can read or write short form of message which is called tweets. The amount of data accumulated on twitter is very huge. This data is unstructured and written in natural language. Twitter Sentimental Analysis is the process of accessing tweets for a particular topic and predicts the sentiment of these tweets as positive, negative or neutral with the help of different machine learning algorithm. The following project is divided into three phases of realisation:-

- Phase 1: This phase analyses sentiments and opinions of people towards the popular politicians like Modi, Rahul Gandhi and Arvind Kejriwal.
- Phase 2: This phase does individual politician analysis and talks about the different politicians, their tweets and analyses their sentiments and popularity with respect to the different tweets that have been generated.
- Phase 3: The last phase is used to measure the deviation and polarity of voters towards the political leaders. This phase will draw a conclusion on who is the most likely candidate for winning the elections based on machine learning approaches.

1.1 Problem Statement

Evaluation of original tweets and updates, how often they are shared and what they are referencing helps to draw conclusions on voter opinions. Similarly, analysing shared content such as likes and comments on Facebook and Twitter etc. helps to determine voter segmentations, where they get their news and opinion from as well as the general mood surrounding elections. Search engine queries help develop an understanding of interest levels shown in the electorate. Twitter is a particularly useful method of gauging voter opinion because it is the most popular social media platform. Approximately 936 million people use it every day. Social Network is an efficient manner to perform electoral research. Attempts can be made to increase classification categories and make the whole process more reliable. Political parties can adopt this method in order to get a better insight into electorates and their policies for a better election performance. The main objective of this project is to perform the sentiment analysis on Indian Political Parties like BJP, INC and AAP along with the leaders that belong to these parties, such that people opinions about these parties progress, workers, policies, etc. which are extracted from Twitter. Thus to achieve this objective we build a classifier based on supervised learning and perform live sentiment analysis on data collected of different political parties. The following project aims at:-

- Mining tweets in real time from twitter
- Performing analysis on tweets, building datasets
- Building a prediction model for predicting the most popular politician/party in the upcoming general elections of 2019
- To build a classifier based on different supervised machine learning techniques.
- Training and testing of build classifier using large datasets
- Comparing results of each classifier and plotting a graph that show the trend of positive and negative sentiment for different political parties.

2. Literature Survey

2.1 Survey of existing models

The following section deals with the various methods that have been followed in the area of text mining and sentiment analysis. Text mining and sentiment analysis find usage in a variety of areas ranging including business intelligence, political science, law/policy making, sociology and psychology. There are a number of techniques that can be used to achieve this. The approach could be supervised or unsupervised learning. For unsupervised learning, the approaches by three different authors. The first approach calculates the semantic orientation of adjectives and verbs in a sentence and determines the overall polarity by adding up the independent values for semantic orientation. This algorithm achieved accuracy of 74% [1]. Another approach uses the Google search engine to define associations for positive and negative words followed by counting the total positive and negative words to determine the overall polarity of a blog [2]. The sentiment analysis is done on a number of levels. The levels are stated below.

1. Phrase Level: This works with phrases of a particular document of analysis. The words which are closer to each other are called as phrases. Phrase level classification is performed on the phrases which contain opinion words. This algorithm cannot be used for finding the long range dependency [3].
2. Aspect Level: Aspect level analysis is a form of feature based classification. Instead on studying structure of the sentence it directly deals with the aspects involved in the text. Based on this analysis, a structured summary of sentiments about products and their features can be produced [4][5].
3. Sentence Level: This is based on the principle of finding out polarity of each sentence. It considers subjectivity and objectivity of the sentence. Subjectivity is related to domain, one sentence is opinion about single domain. The drawback with this is that if the complexity level of the sentence increases, the classification fails. [6][7]
4. Document Level: Works with the documents which contain e-text. Document Level analysis classifies the whole document where sentiments of the single document is

considered so it does not work well with news portal data ,blogs .It is used with both supervised and unsupervised learning algorithms [8][9].

5. Natural Language Processing: Natural Language Processing works with the grammar of the sentence , on the basis of grammar it searches the nouns ,adjectives, verbs, etc. for aspect level classification it is best suited. The disadvantage of Natural Language Processing is it fails if the sentences contain grammatically incorrect words, but we can overcome it by pre-processing the sentences [10][11][12]

The following levels/approaches define how a particular textual entity can be dealt with when performing sentiment analysis. A number of other approaches to sentiment mining have been observed. Another paper uses cross domain sentiment analysis. A two stage architecture/framework described later has been used in another work. The first stage consist of creating an association between the source and the target domain by applying graph ranking algorithms. This is followed by selection of the best seeds from the target domain. In the second stage, these use the essential structure to calculate the sentiment score of each document and then the target domain documents are labelled based on these scores. The latter two phase approach a feature translator was used which could translate a feature in source domain to a target domain feature. In the second stage feature, the source domain data is used to train the classifier and is used to classify the unlabelled data in the target domain [13][14]. These algorithms achieve a decent output, although there is a scope to achieve optimisation regarding the same. One of the first few papers on sentiment/text analysis. The main aim of the paper was to classify text by overall sentiment, not just by topic e.g., classifying movie review either positive or negative. They applied machine learning algorithm on movie review database which results that these algorithms out-perform human produced algorithms. The machine learning algorithms they use are Naïve-Bayes, maximum entropy, and support vector machines. They also conclude by examining various factors that classification of sentiment is very challenging. They show supervised machine learning algorithms are the base for sentiment analysis [15][16]. The choice of the correct classifier for sentiment analysis is also important. A number of techniques have been used in the past. Some famous classifiers are Naïve-Bayes Classifier, Multinomial Naïve Bayes Classifier, Bernoulli Naïve Bayes Classifier, Logistic Regression Classifier, SGDC (Stochastic Gradient Decent Classifier), SVC (Support Vector Classifier): Linear Support Vector Classifier and Nu Support Vector Classifier [17][18][19].

2.2 Summary

A number of techniques have been used for sentiment analysis and a lot of them have given decent results. In the techniques used above, the overall accuracy has been roughly 70% which is less in comparison to supervised learning methods. Thus, it reinforces the claim that an accurate and contextually relevant training data set is vital to achieve highly accurate results for text classification. Thus, the proposed work aims at developing a proper framework to classify sentiments of tweets. The tweet classification is based on a naïve Bayes classifier model. Supervised learning techniques are used to make sense out of the obtained data. The tweets can be used to identify the sentiments of people towards a particular politician and predict the most popular sentiment based on a polarity score.

3. Proposed Method

3.1 Introduction

The project involves a number of phases. The project starts with the first phase that is the collection of data. The project is built on a real time framework, it reads the data and analyses the data in real time using text blob, natural language processing library. The phases involved in the project are as follows:-

1. Stream Data
2. Pre-process data
3. Classification
4. Sentiment Analysis
5. Predictions based on data from social media ranging from multiple parties to various leaders
6. Data visualisation

3.2 Proposed Architecture

3.2.1 Data Collection/ Data Stream

The data is collected from twitter using the twitter data mining API. Twitter Streaming API is used to fetch data relevant to the potential candidates for the Indian election. The Streaming APIs give developers low latency access to Twitters global stream of Tweet data . The input parameters to the streaming functions were the names of the potential candidates like Narendra Modi, Rahul Gandhi, Arvind Kejriwal and other keywords like “Congress”, “BJP”, “AAP” “Right Wing”, “Leftist” etc. Tweets corresponding to the given parameters were returned in JSON format. The JSON result basically comprised of key-value pairs. Some keys were created at, id, re-tweeted, screen name, location etc. The JSON responses were culled to extract only the body of the tweet and stored in a CSV file.

3.2.2 Data Pre-processing

The pre-processing stage consist of the following

1. Removing URLs and pictures
2. Filtering tweets which have candidates name.

3. Hashtags, mentions, and retweets are not removed in order to preserve the original meaning of a tweet.
4. Remove @
5. Remove stop-words before feeding the tweets for classification and prediction.

3.2.3 Data Labelling and analysis

After the data has been cleaned it can be stored in a data frame. The data frame is used to perform analysis on the tweets. The following project uses naïve Bayes classifier in order to perform analysis on tweets. The tweets are given a score from -1<x<1 based on their polarity, where a polarity of -1 is negative and a polarity of 1 is positive. The number of tweets belonging to each polarity is then counted and the data is visualised using python. The figure given below demonstrates the stages involved in realisation of the project.

3.3 Proposed System Model

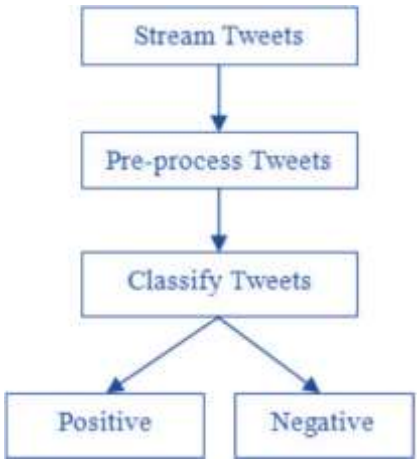


Figure 1

The architecture of the proposed system

4. System Analysis and Design

The proposed system implements a Naive Bayes Classifier makes use of all the features in the feature vector and analyses them individually as they are equally independent of each other. The conditional probability for Naive Bayes can be defined as follows:-

$$P(X \mid y_j) = {}_{i=1}II^{m}\, P(x_i \mid y_j)$$

'X' is the feature vector defined as $X=f(x_1,x_2,....x_m)$ and y_j is the class label. In our project there are different independent features like emoticons, emotional keyword, count of positive and negative keywords, and count of positive and negative hash tags which are effectively utilized by Naïve Bayes classifier for classification. Nave Bayes does not consider the relationships between features. So it cannot utilize the relationships between part of speech tag, emotional keyword and negation.

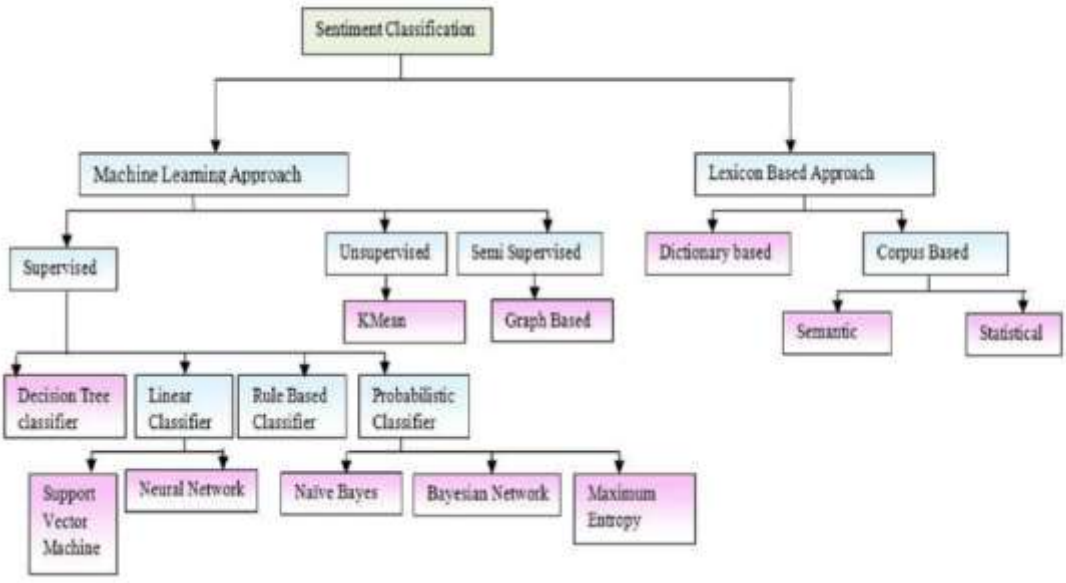


Figure 2

The different methods for doing sentiment analysis

5. Implementation

Languages, frameworks and libraries used in the project:

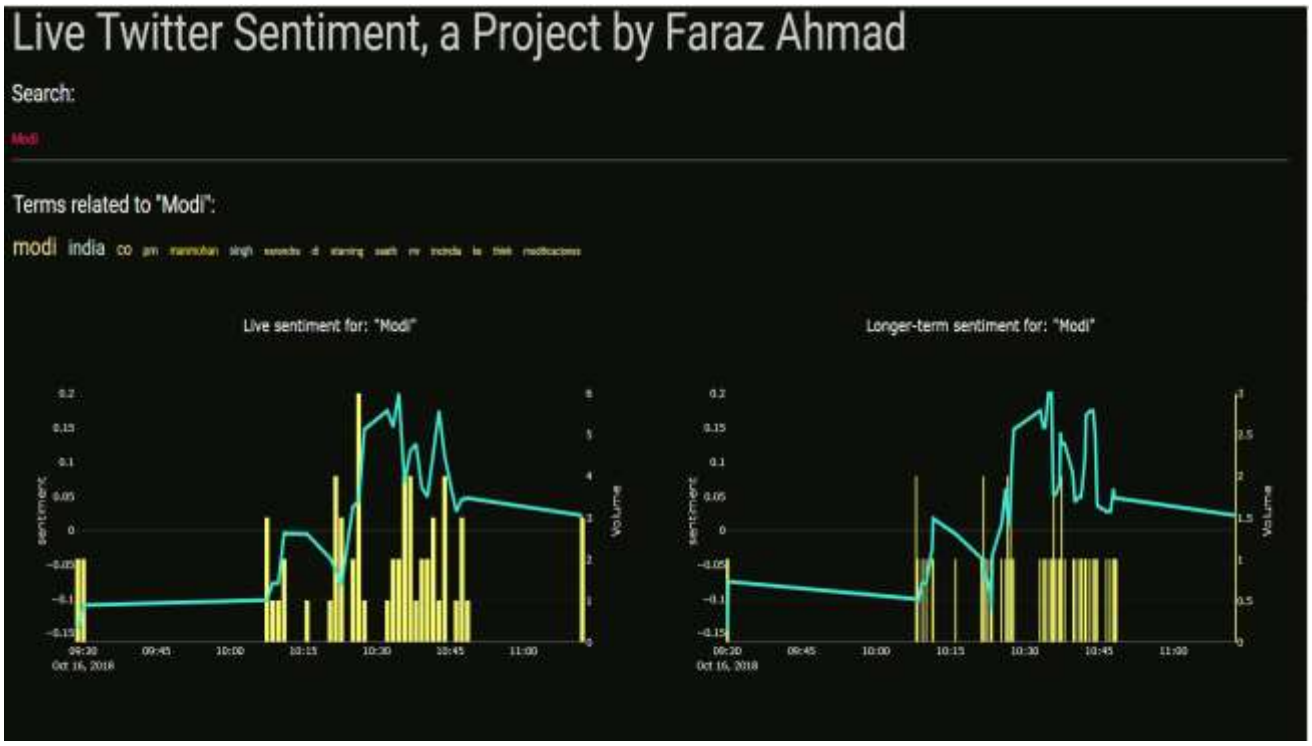
- Python
- Flask
- HTML
- Matplotlib
- Dash
- TextBlob
- Natural Language Processing Toolkit
- Pandas
- Wordcloud
- Scikit learn
- SQLlite

The following flask platform consists of a dynamic page. The page uses a real time graph in order to track the incoming tweets. The tweets generated every second are stored in an SQLlite database. The twitter API generates data in a JSON format which is used by the flask program. The flask program feeds the data to the dash command and is used to generate the graph. There are two views available to the user :-

- Long Term Sentiment Analysis: Gives the sentiment analysis of a particular politician or entity over a period of time. It takes in the data available for a longest timeframe like a week or a month and then plots the graph in real time.
- Live Sentiment Analysis: The live twitter sentiment analysis is real time chart that gets updated every millisecond for the incoming tweets. It analyses the sentiments of a given politician in real time.

Apart from the real time sentiment analysis, the program offers a tweet table which displays the latest trending topics, tweets and hashtags that are relevant to the given topic. The pie-chart on the bottom right is used to display the sentiment wise display of the topic at hand. All the data in the SQLlite table gets updated every second. The program operates in real time thus the

twitter stream has to be active until the program is running. Thus, we are able to accomplish a real time, dynamic and scalable system.



Date	Tweet	Sentiment
2018-10-16 16:42:47.616000	With a view to promote tourism and overall development of these Islands, the Ministry of Home Affairs, Government of India, modifying its earlier orders had removed 29 inhabited Islands of... https://t.co/hItPVqujtT	0.3818
2018-10-16 16:42:44.681000	RT @INCIndia: I want to ask everyone here if Mr. Modi or CM Shivraj Chouhan have done anything for the poor and marginalised sections of Ma...	-0.4215
2018-10-16 16:42:44.683000	RT @RealHistoryPic: "I think that I should change my name from Narendra Modi to Manmohan Singh so that I can again blame Manmohan Singh for...	-0.34
2018-10-16 16:18:42.626000	RT @abhisar_sharma: modiiji pr vidhaansbhaa me gnde aarop (snooping and harrassment) lgaane ke saath saath @KapilMishra_IND ne ye bhii khaa thaa modii kii...	0
2018-10-16 16:18:15.688000	RT @RajaSin38223636: Pm Modi lives life like a King, without knowing that Indians are Starving. #StarvingIndia	-0.0772
2018-10-16 16:18:08.636000	I liked a @YouTube video https://t.co/ArhUcw4kpJ nova modificacao do mod superheroes	0.4215
2018-10-16 16:17:33.624000	Proyecto de Ley. Adenda al Consenso Fiscal. Modificaciones a lo Acordado Respecto al Impuesto a las Ganancias, sobre los Bienes Personales y de Sellos. https://t.co/DUojtIM1VG	0
2018-10-16 16:16:32.504000	Jatin Das has had a tough time ever since he was thrown out of a Lutyens bungalow by the Modi government for overstaying. Cheap Bollywood people.	-0.128
2018-10-16 16:14:44.667000	RT @ModiiRS: WOAH! https://t.co/6xCnTKKLBh	0
2018-10-16 16:14:34.752000	RT @RealHistoryPic: "I think that I should change my name from Narendra Modi to Manmohan Singh so that I can again blame Manmohan Singh for...	-0.34

Figure 3

The Twitter Sentiment Portal on Flask

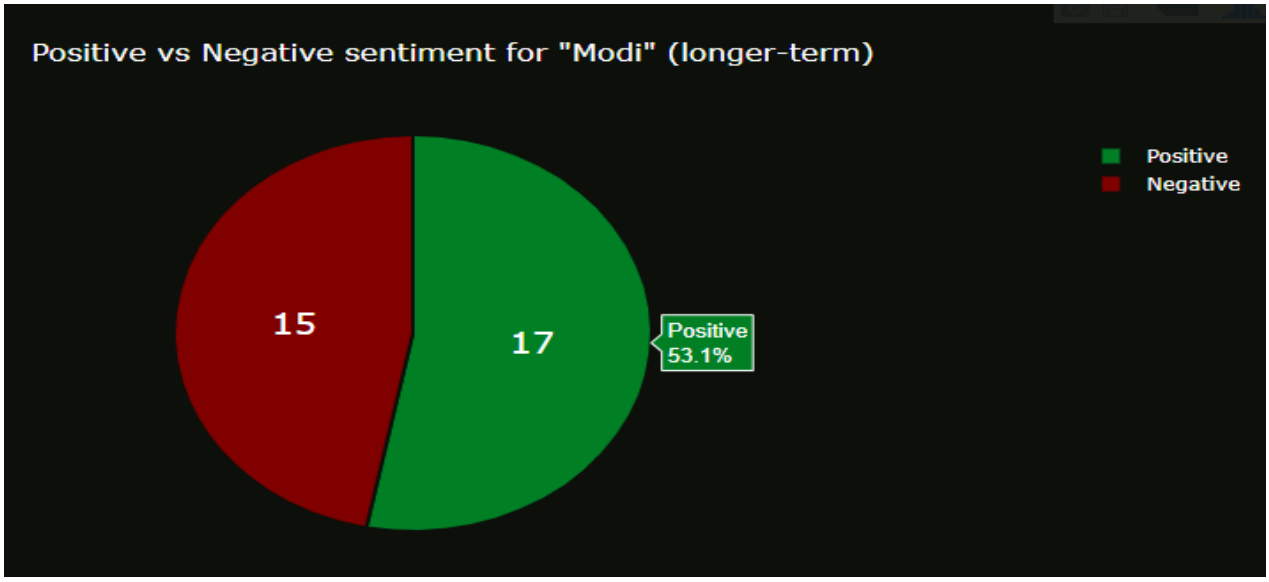


Figure 4

A pie chart showcasing the live sentiments of tweeters

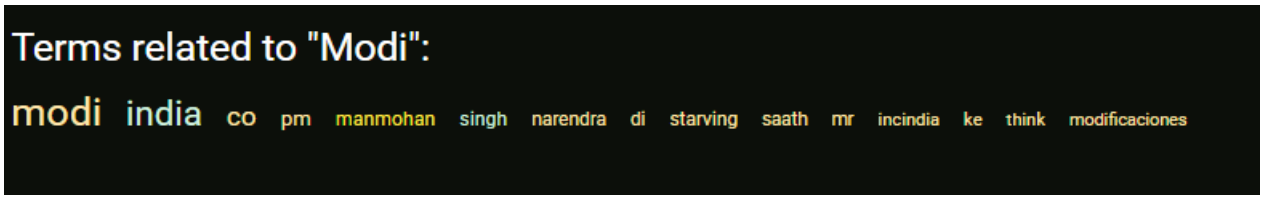


Figure 5

Image showing the terms related to the query

The project serves as an analytics tool which can be used by the political parties and candidates to:-

- Get proper election analytics
- Get idea about the sentiments of people
- Formulate election policies
- Formulate speeches and tweets to get maximum favourability.

Generation of a word cloud: A word cloud is a powerful visualisation tool that is used to find the most used words in a given bag of words. It operates based on frequency and relevance. The images given below shows a word cloud for the three politicians, Modi, Kejriwal and Rahul Gandhi.

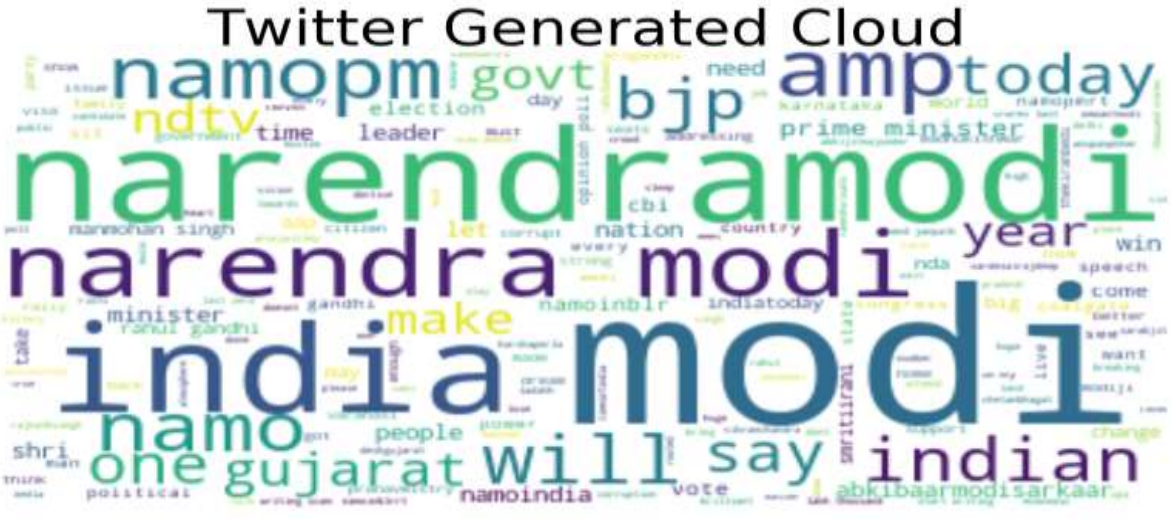


Figure 6

The word cloud for Narendra Modi

Figure 7

The word cloud for Arvind Kejriwal

Figure 8

The word cloud for Rahul Gandhi

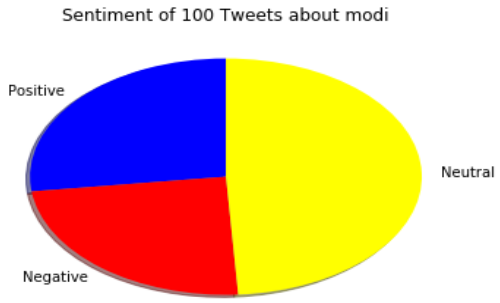
6. Results and Discussions

The project was successfully completed and the different analysis were performed. The analysis of results could be divided into three phases.

Phase 1: Individual Politician Analysis

This phase accomplished a general program which can be used to find the recent sentiments of people on twitter on a given topic. The number of tweets can be stated by the user to perform the analysis.

```
In [17]: runfile('C:/Users/FARAZ AHMAD/Desktop/Political
analysis/Twitter-Sentiment-Analysis Faraz/
sentimentanalysisfaraz.py', wdir='C:/Users/FARAZ AHMAD/Desktop/
Political analysis/Twitter-Sentiment-Analysis Faraz')

Enter the topic you want to search about
modi

Enter the number of tweets to search
100

The tweet with most likes is:
Dr Manmohan Singh hardly spoke unless it mattered. PM Modi
hardly speaks when it matters. I know which I'd rather have.
What about you?
Number of likes: 10359
135 characters.

The tweet with most retweets is:
Dr Manmohan Singh hardly spoke unless it mattered. PM Modi
hardly speaks when it matters. I know which I'd rather have.
What about you?
Number of retweets: 3012
135 characters.
```



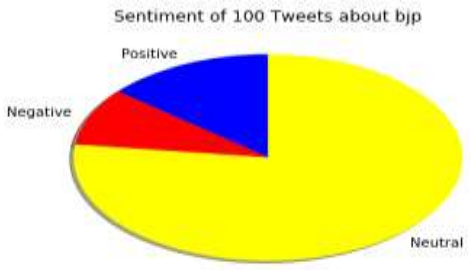

```
Percentage of positive tweets: 27.0 %

Percentage of negative tweets: 24.0 %

Percentage of neutral tweets: 49.0 %
```

Figure showing the sentiment of people towards the topic "modi"

	Tweets	len	ID	Date	Source	Likes	RTs	Sentiment	Senti
0	What hap	140	1.05E+18	########	Twitter for	1777	891	Neutral	0
1	In Mamat	140	1.05E+18	########	Twitter for	598	815	Neutral	0
2	I feel	139	1.05E+18	########	Twitter for	775	428	Negative	-1
3	India's Dis	115	1.05E+18	########	Twitter for	0	0	Neutral	0
4	The chief	128	1.05E+18	########	Twitter for	0	0	Positive	1
5	Modi's #Ir	89	1.05E+18	########	Twitter for	1	0	Neutral	0
6	Modi Govt	140	1.05E+18	########	Twitter W	0	0	Negative	-1
7	"PM Modi	126	1.05E+18	########	IFTTT	0	0	Neutral	0
8	Amidst thr	132	1.05E+18	########	Twitter W	0	0	Neutral	0
9	@narend	37	1.05E+18	########	Twitter for	0	0	Neutral	0
10	'PM Modi	139	1.05E+18	########	Twitter for	0	0	Negative	-1
11	..and now	88	1.05E+18	########	Twitter for	0	0	Neutral	0
12	@Swapan	140	1.05E+18	########	Twitter W	0	0	Neutral	0

Data Collection is done in the form of a csv file which can be used later on for analysis and predictions

```
Enter the topic you want to search about
bjp

Enter the number of tweets to search
100
```




```
Percentage of positive tweets: 14.000000000000002 %

Percentage of negative tweets: 9.0 %

Percentage of neutral tweets: 77.0 %
```

Another twitter sentiment analysis for the topic named “bjp”

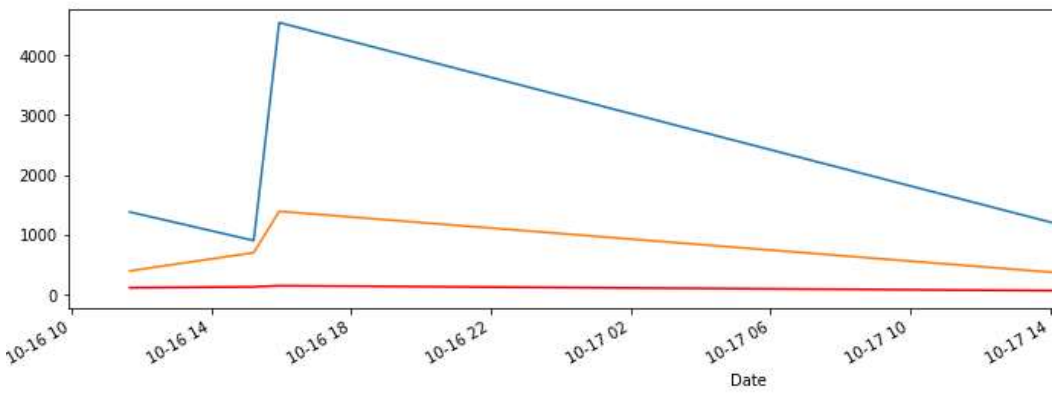


Figure 10

A real time activity graph

The program written in phase 1 was a general program which could be optimised to suit any input criteria and find the sentiment of the same.

Phase 2: Individual Politician Analysis

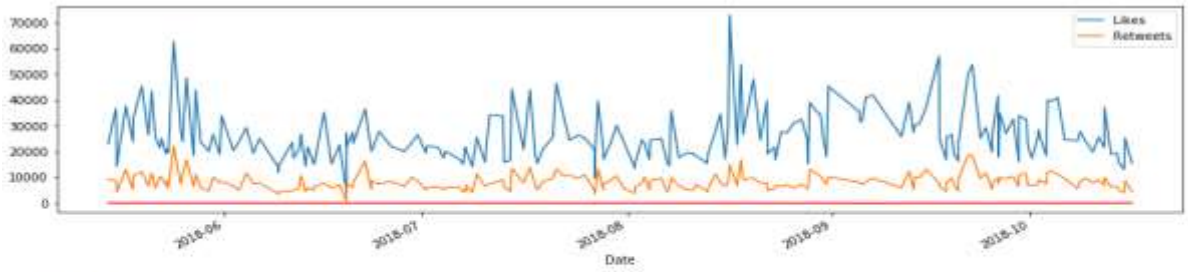
This phase aimed at finding out the most popular politician, and the activity of people on the recent tweets by the politician. The activities taking place in this phase are:-

- Analysing the last 10 tweets of a particular politician
- Analysing the tweets which got the most likes
- Analysing the tweets which got the most shares/retweets
- Drawing a favourite/retweet real time graph to get a better idea about the activity
- Finding the media used by the politician to tweet

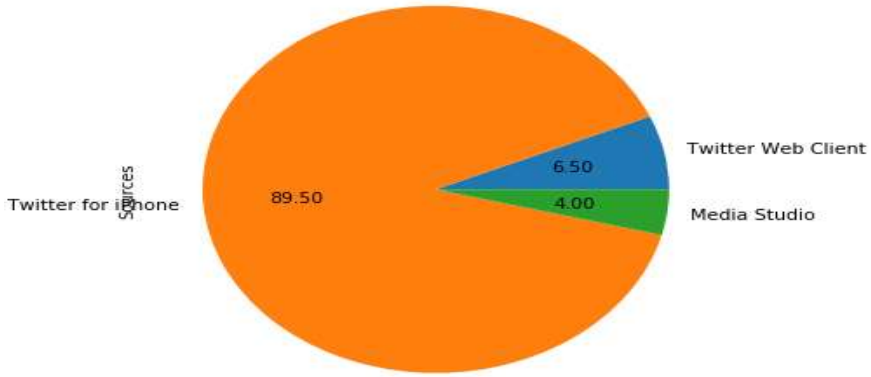
- **Popularity Analysis of Rahul Gandhi**

	ID	Date	Source	Likes	RTs
0	1052134915079266304	2018-10-16 09:51:26	Twitter Web Client	15550	4552
1	1051759663308521472	2018-10-15 09:00:19	Twitter for iPhone	25312	8646
2	1051699305969590272	2018-10-15 05:00:28	Twitter for iPhone	13123	4034
3	1051415198509207552	2018-10-14 10:11:32	Twitter for iPhone	15892	5227
4	1051327932105404416	2018-10-14 04:24:46	Media Studio	19302	6309
5	1050983130536927234	2018-10-13 05:34:39	Twitter for iPhone	19061	6402
6	1050629984014884864	2018-10-12 06:11:22	Twitter for iPhone	37125	10051
7	1050609724800913409	2018-10-12 04:50:52	Twitter for iPhone	21602	6808
8	1050311182593286145	2018-10-11 09:04:34	Twitter for iPhone	25715	9089
9	1049992547765035010	2018-10-10 11:58:25	Twitter for iPhone	19822	7866

```
The lenght's average in tweets: 135.645
The tweet with more likes is:
Today India lost a great son.  Former PM, Atal Bihari Vajpayee ji, was loved and
respected by millions. My condolen… https://t.co/ak5QL5V7Qp
Number of likes: 72943
140 characters.

The tweet with more retweets is:
Dear PM,

Glad to see you accept the @imVkohli fitness challenge. Here’s one from me:

Reduce Fuel prices or the C… https://t.co/pLoH6A6Cp2
Number of retweets: 22221
140 characters.
```




```
Creation of content sources:
* Twitter Web Client
* Twitter for iPhone
* Media Studio
```



Popularity analysis of Rahul Gandhi with insights into his tweets

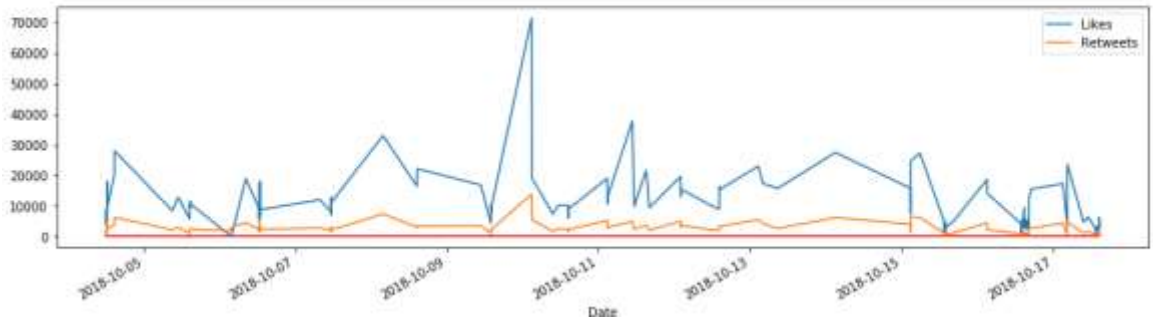
- **Popularity Analysis of Narendra Modi**

```
                   ID                 Date              Source  Likes   RTs
0  1052568954772840449 2018-10-17 14:36:09       Media Studio   3401   995
1  1052568682503786496 2018-10-17 14:35:04       Media Studio   4404  1327
2  1052568515901837313 2018-10-17 14:34:24       Media Studio   4112  1066
3  1052568207410769920 2018-10-17 14:33:11       Media Studio   6254  1646
4  1052567873288331264 2018-10-17 14:31:51       Media Studio   2289   695
5  1052567730627457026 2018-10-17 14:31:17       Media Studio   2862   831
6  1052567616055926785 2018-10-17 14:30:50       Media Studio   3612  1271
7  1052553234412273664 2018-10-17 13:33:41  Twitter Web Client   1404   403
8  1052553231463661568 2018-10-17 13:33:40  Twitter Web Client   3182   743
9  1052552857046548481 2018-10-17 13:32:11  Twitter Web Client   1094   301
starting visualisation...


The lenght's average in tweets: 134.42
The tweet with more likes is:
नवरात्रि के पावन पर्व पर हार्दिक शुभकामनाएं

शक्ति की देवी मां दुर्गा सबके जीवन में सुख, शांति और समृद्धि लेकर आएं

जय मां जगदम्बा!
Number of likes: 71455
135 characters.
```

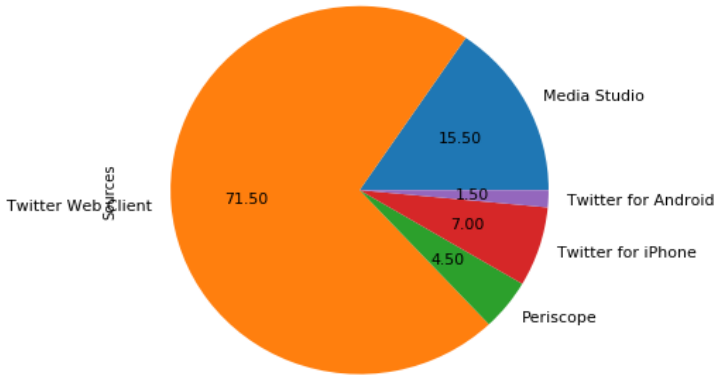
The tweet with more retweets is:
नवरात्रि के पावन पर्व पर हार्दिक शुभकामनाएं

शक्ति की देवी मां दुर्गा सबके जीवन में सुख, शांति और समृद्धि लेकर आएं

जय मां जगदम्बा!
Number of retweets: 13808
135 characters.



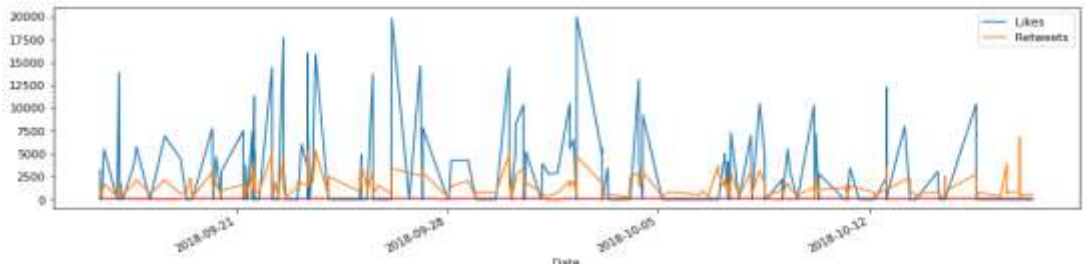
Creation of content sources:
* Media Studio
* Twitter Web Client
* Periscope
* Twitter for iPhone
* Twitter for Android



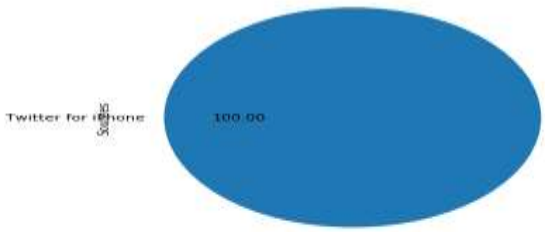
The popularity analysis of Narendra Modi along with insights into the tweets

- **Popularity Analysis of Arvind Kejriwal**

```
                   ID                 Date             Source  Likes   RTs
0  1052499379209293824  2018-10-17 09:59:41  Twitter for iPhone      0   611
1  1052346089045680129  2018-10-16 23:50:33  Twitter for iPhone      0   483
2  1052345649868476416  2018-10-16 23:48:49  Twitter for iPhone      0   136
3  1052345617295466496  2018-10-16 23:48:41  Twitter for iPhone      0   665
4  1052341112650825731  2018-10-16 23:30:47  Twitter for iPhone      0  6876
5  1052339756015730690  2018-10-16 23:25:24  Twitter for iPhone      0   865
6  1052206899255373827  2018-10-16 14:37:28  Twitter for iPhone      0   806
7  1052190886069657600  2018-10-16 13:33:50  Twitter for iPhone      0   954
8  1052190859733549056  2018-10-16 13:33:44  Twitter for iPhone      0   607
9  1052190697439150080  2018-10-16 13:33:05  Twitter for iPhone      0  3943
starting visualisation...


The lenght's average in tweets: 126.34
The tweet with more likes is:
दिल्ली सबकी है  किसानों को दिल्ली में आने से नहीं रोका जा सकता  किसानों की माँगे जायज़ हैं उनकी माँगें मानी जायें
Number of likes: 19969
115 characters.

The tweet with more retweets is:
RT @faridque: How to face tough questions from #journalists 😂😂 https://
t.co/D1HnC2pt0M
Number of retweets: 6876
86 characters.
```




```
Creation of content sources:
* Twitter for iPhone
```



The popularity analysis of Kejriwal with tweets insights

Phase 3: The prediction and community detection

Finally, the most cumbersome task was to use our classifiers and algorithm to predict who is in line for being the next Prime Minister of India. For this we used a generalisation algorithm based on the measure of polarity. K-means algorithm and Hierarchical clustering was used to see the trends and opinions of people and form communities. Supporters which tweeted positively for BJP were found to tweet negatively against the opposition parties. Thus, these people formed a group/community which could be called the BJP supporters, this could be applied to supporters of AAP and Congress as well. The other community consisted of neutral sentiments which basically had no political affiliations or deviations. They formed the centroid of the clusters. The final result was then fed to the classifiers and results were obtained in real time. **It is important to note that since the results are based on the real time sentiment of the people, they can be expected to change anytime.**

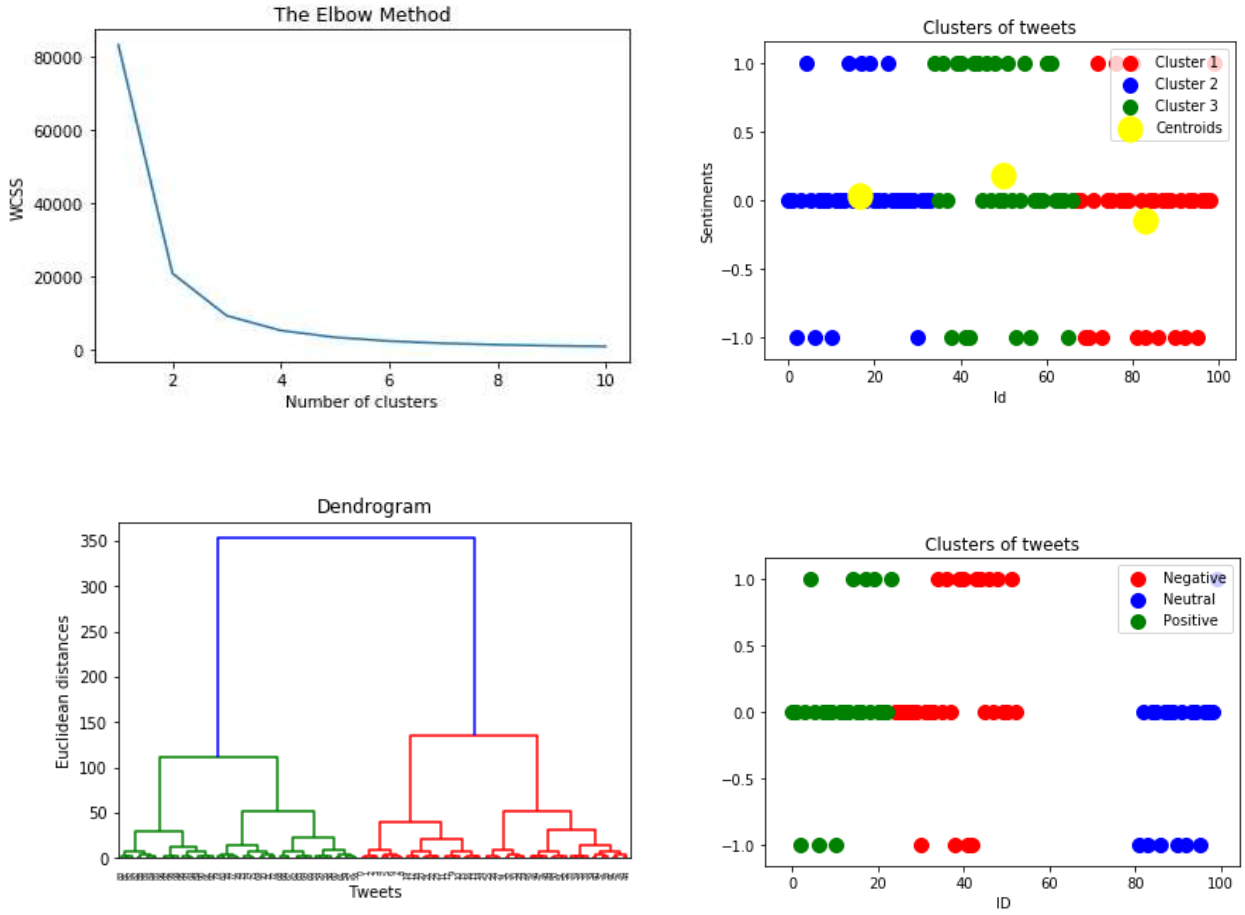


Figure 11 (a) K-means clustering

(b) Hierarchical clustering

In the final stage, the clustering used to identify the communities that are involved in the following analysis. The communities were formed by supporters of different parties The clustering index was used as a reference to find the polarity score and thus predict the most important question, which party is going to win the general elections of 2019. The algorithm achieved an accuracy of 85%.

```
In [21]: runfile('C:/Users/FARAZ AHMAD/.spyder-py3/
prediction.py', wdir='C:/Users/FARAZ AHMAD/.spyder-py3')
Getting tweets...
As of now congress has a higher polarity score and can be
expected to win!
0.13428571428571429
```

The final prediction

Note: Since the results are based on the real time sentiment of the people, they can be expected to change anytime.

7. Conclusion and Future Scope

Sentiment analysis is used to identifying people's opinion, attitude and emotional states. The views of the people can be positive or negative. Commonly, parts of speech are used as feature to extract the sentiment of the text. An adjective plays a crucial role in identifying sentiment from parts of speech. Sometimes words having adjective and adverb are used together then it is difficult to identify sentiment and opinion. To do the sentiment analysis of tweets, the proposed system first extracts the twitter posts from twitter by user. The system can also computes the frequency of each term in tweet. Using machine learning supervised approach help to obtain the results. Twitter is large source of data, which make it more attractive for performing sentiment analysis. We perform analysis on around 15,000 tweets total for each party, so that we analyse the results, understand the patterns and give a review on people opinion. We saw different party have different sentiment results according to their progress and working procedure. We also saw how any social event, speech or rally cause a fluctuation in sentiment of people. We also get to know which policies are getting more support from people which are started by any of these parties. It was shown that BJP is more successful political part in present time based on people opinion. It is not necessary that our classifier can only be used for political parties. It is general classifier. It can be used for any purpose based on tweets we collect with the help of keyword. It can be used for finance, marketing, reviewing and many more. The future improvements for the systems can include the following:-

- Use of parser can be embedded into system to improve results.
- We can improve our system that can deal with sentences of multiple meanings.
- We can also increase the classification categories so that we can get better results.
- We can start work on multi languages like Hindi, Spanish, and Arabic to provide sentiment analysis to more local.
- Working on detecting sarcasm in speech is another challenge.

8. References

[1] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," presented at the Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania,2002.

[2] A. Harb, M. Planti, G. Dray, M. Roche, Fran, o. Trousset, and P.Poncelet, "Web opinion mining: how to extract opinions from blogs?," presented at the Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology, Cergy-Pontoise, France, 2008.

[3] Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. "Recognizing contextual polarity in phrase-level sentiment analysis." Proceedings of the conference on human language technology and empirical methods in natural language processing. Association for Computational Linguistics, 2005.

[4] Singh, Vivek Kumar, et al. "Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification." Automation, computing, communication, control and compressed sensing (iMac4s), 2013 international multi-conference on. IEEE, 2013.

[5] Schouten, Kim, and Flavius Frasincar. "Survey on aspect-level sentiment analysis." IEEE Transactions on Knowledge & Data Engineering 1 (2016): 1-1.

[6] Wang, Dingding, et al. "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization." Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2008.

[7] Pang, Bo, and Lillian Lee. "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts." Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2004.

[8] Zhang, Changli, et al. "Sentiment analysis of Chinese documents: From sentence to document level." Journal of the American Society for Information Science and Technology 60.12 (2009): 2474-2487.

[9] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends in Information Retrieval 2.1–2 (2008): 1-135.

[10] Liddy, Elizabeth D. "Natural language processing." (2001).

[11] Jurafsky, Daniel. "Speech and language processing: An introduction to natural language processing." Computational linguistics, and speech recognition (2000).

[12] Manning, Christopher, et al. "The Stanford Core NLP natural language processing toolkit." Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. 2014.

[13] K. Liu and J. Zhao, "Cross-domain sentiment classification using atwostage method," presented at the Proceedings of the 18th ACM conference on Information and knowledge management, Hong Kong, China, 2009.

[14] Q. Wu, S. Tan, H. Zhai, G. Zhang, M. Duan, and X. Cheng,"Senti Rank: Cross-Domain Graph Ranking for Sentiment Classification," presented at the Proceedings of the 2009IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01, 2009.

[15] P. Pang, L. Lee and S. Vaidyanathan, “Thumbs up? sentiment classification using machine learning techniques”, Proc. ACL-02 conference on Empirical methods in natural language processing, vol.10, pp. 79-86, 2002.

[16]A. Pak and P. Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining”, vol. 10, pp. 1320-1326, 2010.

[17] Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." Neural processing letters 9.3 (1999): 293-300.

[18] Dumais, Susan T., et al. "Methods and apparatus for classifying text and for building a text classifier." U.S. Patent No. 6,192,360. 20 Feb. 2001.

[19] Ng, Andrew Y., and Michael I. Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." Advances in neural information processing systems. 2002.

Appendix

Appendix A: General Code for Sentiment Analysis

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep  3 09:47:19 2018

@author: FARAZ AHMAD
"""
import tweepy
from textblob import TextBlob
import csv
import re
import sys
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display
#Does plotting as well as analysis
consumer_key=''
consumer_secret=''

access_token_key=''
access_token_secret=''

auth=tweepy.OAuthHandler(consumer_key,consumer_secret)
auth.set_access_token(access_token_key,access_token_secret)

api=tweepy.API(auth)
```

```
topicname=input("Enter the topic you want to search about\n")
number=int(input("Enter the number of tweets to search\n"))
public_tweets=api.search(
   lang="en",
   q=topicname + " -rt",
   count=number, result_type='mixed')
#result_type=mixed,recent,poular (Faraz Ahmad edited)
unwanted_words=['@','RT',':','https','http']
symbols=['@','#']
data=[]
data = pd.DataFrame(data=[tweet.text for tweet in public_tweets], columns=['Tweets'])
display(data.head(number))
data['len']  = np.array([len(tweet.text) for tweet in public_tweets])
data['ID']   = np.array([tweet.id for tweet in public_tweets])
data['Date'] = np.array([tweet.created_at for tweet in public_tweets])
data['Source'] = np.array([tweet.source for tweet in public_tweets])
data['Likes']  = np.array([tweet.favorite_count for tweet in public_tweets])
data['RTs']    = np.array([tweet.retweet_count for tweet in public_tweets])
display(data.head(number))
fav_max = np.max(data['Likes'])
rt_max  = np.max(data['RTs'])

fav = data[data.Likes == fav_max].index[0]
rt  = data[data.RTs == rt_max].index[0]

# Max FAVs:
print("The tweet with most likes is: \n{}".format(data['Tweets'][fav]))
print("Number of likes: {}".format(fav_max))
print("{} characters.\n".format(data['len'][fav]))

# Max RTs:
```

```
print("The tweet with most retweets is: \n{}".format(data['Tweets'][rt]))
print("Number of retweets: {}".format(rt_max))
print("{} characters.\n".format(data['len'][rt]))
tlen = pd.Series(data=data['len'].values, index=data['Date'])
tfav = pd.Series(data=data['Likes'].values, index=data['Date'])
tret = pd.Series(data=data['RTs'].values, index=data['Date'])
tlen.plot(figsize=(16,4), color='r');
tfav.plot(figsize=(16,4), label="Likes", legend=True)
tret.plot(figsize=(16,4), label="Retweets", legend=True);
plt.show()
countpos=countneg=countneut=0
arr=[]
a1=[]
for tweet in public_tweets:
    tweeta = tweet.text
    tidy_tweet = (tweeta.strip().encode('ascii', 'ignore')).decode("utf-8")
    #tidy_tweet='    '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z    \t])|(\w+:\/\/\S+)|(RT)",    "
",tidy_tweet).split())
    #print(tidy_tweet)
    #print(cleanedTweet)
    analysis= TextBlob(tidy_tweet)
    #print (analysis.sentiment)
    if(analysis.sentiment.polarity > 0.2):
        polarity = 'Positive'
        countpos=countpos+1
        pol=1
    elif(0<=analysis.sentiment.polarity <=0.2):
        polarity = 'Neutral'
        countneut=countneut+1
        pol=0
    elif(analysis.sentiment.polarity < 0):
```

```
        polarity = 'Negative'
        countneg=countneg+1
        pol=-1

    #dic={}
    arr.append(polarity)
    a1.append(pol)
    #df=pd.DataFrame(arr)
    #data['Sentiment']=pd.Series(polarity)
    #dic['Tweet']=tidy_tweet
    #data.append(dic)
se=pd.Series(arr)
df=pd.DataFrame(data)
df['Sentiment']=se.values
ss=pd.Series(a1)
df['Senti']=ss.values
#print(df)

df.to_csv('analysedfile.csv')
positive = countpos
negative = countneg
neutral = countneut
colors = ['blue', 'red', 'yellow']
sizes = [positive, negative, neutral]
labels = 'Positive', 'Negative', 'Neutral'
plt.pie(
   x=sizes,
   shadow=True,
   colors=colors,
   labels=labels,
   startangle=90
```

```
)

plt.title("Sentiment of {} Tweets about {}".format(number, topicname))
plt.show()
percpos=(countpos/int(number))*100
percneg=(countneg/int(number))*100
percneut=(countneut/int(number))*100
print("\nPercentage of positive tweets:",percpos,"%")
print("\nPercentage of negative tweets:",percneg,"%")
print("\nPercentage of neutral tweets:",percneut,"%")
```

Appendix B- Code for politician popularity analysis

```
# -*- coding: utf-8 -*-
"""
Created on Sat Sep  1 02:52:19 2018

@author: FARAZ AHMAD
"""

# General:
import tweepy
import pandas as pd
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import seaborn as sns
#matplotlib inline
CONSUMER_KEY = "hzNYHN9xRlXeKu7g2aj7nWNAI"
CONSUMER_SECRET                                                    =
"Xm3ScyKHRL5EBXdr08n1IHuJjO3YLv1ea68Td5rVCVo56SsYNq"
```

```
ACCESS_TOKEN = "78845728-kLsebXB9e0WCdRMxISdaIRbx2pNzgUzsrNSKSbYDy"
ACCESS_SECRET = "jd7cOjgDi0rysph8kznb4pqofoA0TqFtlUs1RmCJBCPsf"
from credentials import *
def twitter_setup():
    """
    Utility function to setup the Twitter's API
    with our access keys provided.
    """
    # Authentication and access using keys:
    auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
    auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

    # Return API with authentication:
    api = tweepy.API(auth)
    return api
# We create an extractor object:
extractor = twitter_setup()

# We create a tweet list as follows:
tweets = extractor.user_timeline(screen_name="ArvindKejriwal", count=200)
print("Number of tweets extracted: {}.\n".format(len(tweets)))

# We print the most recent 5 tweets:
print("5 recent tweets:\n")
for tweet in tweets[:5]:
    #print(tweet.text)
    print()
# We create a pandas dataframe as follows:
data = pd.DataFrame(data=[tweet.text for tweet in tweets], columns=['Tweets'])

# We display the first 10 elements of the dataframe:
```

```
display(data.head(10))
data['len']  = np.array([len(tweet.text) for tweet in tweets])
data['ID']   = np.array([tweet.id for tweet in tweets])
data['Date'] = np.array([tweet.created_at for tweet in tweets])
data['Source'] = np.array([tweet.source for tweet in tweets])
data['Likes']  = np.array([tweet.favorite_count for tweet in tweets])
data['RTs']    = np.array([tweet.retweet_count for tweet in tweets])
display(data.head(10))
print("starting visualisation...\n\n")
mean = np.mean(data['len'])

print("The lenght's average in tweets: {}".format(mean))
fav_max = np.max(data['Likes'])
rt_max  = np.max(data['RTs'])

fav = data[data.Likes == fav_max].index[0]
rt  = data[data.RTs == rt_max].index[0]

# Max FAVs:
print("The tweet with more likes is: \n{}".format(data['Tweets'][fav]))
print("Number of likes: {}".format(fav_max))
print("{} characters.\n".format(data['len'][fav]))

# Max RTs:
print("The tweet with more retweets is: \n{}".format(data['Tweets'][rt]))
print("Number of retweets: {}".format(rt_max))
print("{} characters.\n".format(data['len'][rt]))
tlen = pd.Series(data=data['len'].values, index=data['Date'])
tfav = pd.Series(data=data['Likes'].values, index=data['Date'])
tret = pd.Series(data=data['RTs'].values, index=data['Date'])
tlen.plot(figsize=(16,4), color='r');
```

```
tfav.plot(figsize=(16,4), label="Likes", legend=True)
tret.plot(figsize=(16,4), label="Retweets", legend=True);
plt.show()
sources = []
for source in data['Source']:
    if source not in sources:
        sources.append(source)

# We print sources list:
print("Creation of content sources:")
for source in sources:
    print("* {}".format(source))



percent = np.zeros(len(sources))

for source in data['Source']:
    for index in range(len(sources)):
        if source == sources[index]:
            percent[index] += 1
            pass

percent /= 100

# Pie chart:
pie_chart = pd.Series(percent, index=sources, name='Sources')
pie_chart.plot.pie(fontsize=11, autopct='%.2f', figsize=(6, 6));
```

Appendix C- Code for prediction

```
"""
@author Faraz Ahmad
"""

#twitter API
import tweepy
#sentiment analysis library
from textblob import TextBlob

#get these from https://apps.twitter.com/

consumer_key='hzNYHN9xRlXeKu7g2aj7nWNAI'
consumer_secret='Xm3ScyKHRL5EBXdr08n1IHuJjO3YLv1ea68Td5rVCVo56SsYNq'

access_token='78845728-kLsebXB9e0WCdRMxISdaIRbx2pNzgUzsrNSKSbYDy'
access_token_secret='jd7cOjgDi0rysph8kznb4pqofoA0TqFtlUs1RmCJBCPsf'


#initializing twitter
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

congress_polarity = []
bjp_polarity = []


def analyze(tweet_text, person):
    if(person == "congress"):
```

```
        analysis = TextBlob(tweet_text)
        congress_polarity.append(analysis.sentiment[0])
    elif(person == "bjp"):
        analysis = TextBlob(tweet_text)
        bjp_polarity.append(analysis.sentiment[0])



def main():
    print ("Getting tweets...")
    for i in ["congress", "bjp"]:
        for tweet in api.search(i):
            analyze(tweet.text, i)

    cl_avg, bjp_avg = sum(congress_polarity)/len(congress_polarity), sum(bjp_polarity)/len(bjp_polarity)

    if cl_avg > bjp_avg:
        print ("As of now congress has a higher polarity score and can be expected to win!")
        print(sum(congress_polarity)/len(congress_polarity))
    elif bjp_avg > cl_avg:
        print ("As of now BJP has a higher polarity score")
        print( sum(bjp_polarity)/len(bjp_polarity))
    else:
        print ("The sentiment is neutral for the two parties, rise of another block can be expected")



if __name__ == '__main__':
    main()
# K-Means Clustering
#Faraz Ahmad
# Importing the libraries
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('analysedfile.csv')
X = dataset.iloc[:, [0, 9]].values
# y = dataset.iloc[:, 3].values

# Splitting the dataset into the Training set and Test set
"""from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""

# Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
```

```
plt.ylabel('WCSS')
plt.show()

# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
#plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
#plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Cluster
5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow',
label = 'Centroids')
plt.title('Clusters of tweets')
plt.xlabel('Id')
plt.ylabel('Sentiments')
plt.legend()
plt.show()
# Hierarchical Clustering

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.cluster.hierarchy as sch
# Importing the dataset
dataset = pd.read_csv('analysedfile.csv')
X = dataset.iloc[:, [0, 9]].values
```

```
# y = dataset.iloc[:, 3].values

# Splitting the dataset into the Training set and Test set
"""from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)"""

# Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""

# Using the dendrogram to find the optimal number of clusters

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Tweets')
plt.ylabel('Euclidean distances')
plt.show()

# Fitting Hierarchical Clustering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)

# Visualising the clusters
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Negative')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Neutral')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Positive')
```

```
#plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
#plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of tweets')
plt.xlabel('ID')
plt.ylabel('Tweets')
plt.legend()
plt.show()
```

Appendix D- Code for the front end

```
# set chdir to current dir
import os
import sys
sys.path.insert(0, os.path.realpath(os.path.dirname(__file__)))
os.chdir(os.path.realpath(os.path.dirname(__file__)))

import dash
from dash.dependencies import Output, Event, Input
import dash_core_components as dcc
import dash_html_components as html
import plotly
import plotly.graph_objs as go
import sqlite3
import pandas as pd

from collections import Counter
import string
import regex as re
from cache import cache
from config import stop_words
import time
```

```
import pickle

# it's ok to use one shared sqlite connection
# as we are making selects only, no need for any kind of serialization as well
conn = sqlite3.connect('twitter.db', check_same_thread=False)

punctuation = [str(i) for i in string.punctuation]



sentiment_colors = {-1:"#EE6055",
                    -0.5:"#FDE74C",
                     0:"#FFE6AC",
                     0.5:"#D0F2DF",
                     1:"#9CEC5B",}


app_colors = {
    'background': '#0C0F0A',
    'text': '#FFFFFF',
    'sentiment-plot':'#41EAD4',
    'volume-bar':'#FBFC74',
    'someothercolor':'#FF206E',
}

POS_NEG_NEUT = 0.1

MAX_DF_LENGTH = 100

app = dash.Dash(__name__)
app.layout = html.Div(
```

```
[    html.Div(className='container-fluid', children=[html.H2('Live Twitter Sentiment, a Project by Faraz Ahmad', style={'color':"#CECECE"}),
                                                     html.H5('Search:', style={'color':app_colors['text']}),
                                                    dcc.Input(id='sentiment_term', value='Modi', type='text', style={'color':app_colors['someothercolor']}),
                                                    ],
                style={'width':'98%','margin-left':10,'margin-right':10,'max-width':50000}),



    html.Div(className='row', children=[html.Div(id='related-sentiment', children=html.Button('Loading related terms...', id='related_term_button'), className='col s12 m6 l6', style={"word-wrap":"break-word"}),
                                        html.Div(id='recent-trending', className='col s12 m6 l6', style={"word-wrap":"break-word"})]),

    html.Div(className='row', children=[html.Div(dcc.Graph(id='live-graph',animate=False), className='col s12 m6 l6'),
                                        html.Div(dcc.Graph(id='historical-graph',animate=False), 
                                        html.Div(id="recent-tweets-table", 
                                        html.Div(dcc.Graph(id='sentiment-pie', animate=False), 
                                        dcc.Interval(
                                            id='graph-update',
                                            interval=11000
                                        ),
                                        dcc.Interval(
                                            id='historical-update',
```

```
            interval=60*1000
        ),

        dcc.Interval(
            id='related-update',
            interval=30*1000
        ),

        dcc.Interval(
            id='recent-table-update',
            interval=2*1000
        ),

        dcc.Interval(
            id='sentiment-pie-update',
            interval=60*1000
        ),

    ], style={'backgroundColor': app_colors['background'], 'margin-top':'-30px',
'height':'2000px',},
)


def df_resample_sizes(df, maxlen=MAX_DF_LENGTH):
    df_len = len(df)
    resample_amt = 100
    vol_df = df.copy()
    vol_df['volume'] = 1

    ms_span = (df.index[-1] - df.index[0]).seconds * 1000
    rs = int(ms_span / maxlen)
```

```
    df = df.resample('{}ms'.format(int(rs))).mean()
    df.dropna(inplace=True)

    vol_df = vol_df.resample('{}ms'.format(int(rs))).sum()
    vol_df.dropna(inplace=True)

    df = df.join(vol_df['volume'])

    return df

# make a counter with blacklist words and empty word with some big value - we'll use it later to filter counter
stop_words.append('')
blacklist_counter = Counter(dict(zip(stop_words, [1000000]*len(stop_words))))

# complie a regex for split operations (punctuation list, plus space and new line)
split_regex = re.compile("[ \n"+re.escape("".join(punctuation))+']')

def related_sentiments(df, sentiment_term, how_many=15):
    try:

        related_words = {}

        # it's way faster to join strings to one string then use regex split using your punctuation list plus space and new line chars
        # regex precomiled above
        tokens = split_regex.split(' '.join(df['tweet'].values.tolist()).lower())

        # it is way faster to remove stop_words, sentiment_term and empty token by making another counter
```

```
        # with some big value and substracting (counter will substract and remove tokens with negative count)
        blacklist_counter_with_term = blacklist_counter.copy()
        blacklist_counter_with_term[sentiment_term] = 1000000
        counts = (Counter(tokens) - blacklist_counter_with_term).most_common(15)

        for term,count in counts:
            try:
                df = pd.read_sql("SELECT sentiment.* FROM  sentiment_fts fts LEFT JOIN sentiment ON fts.rowid = sentiment.id WHERE fts.sentiment_fts MATCH ? ORDER BY fts.rowid DESC LIMIT 200", conn, params=(term,))
                related_words[term] = [df['sentiment'].mean(), count]
            except Exception as e:
                with open('errors.txt','a') as f:
                    f.write(str(e))
                    f.write('\n')

        return related_words

    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')


def quick_color(s):
    # except return bg as app_colors['background']
    if s >= POS_NEG_NEUT:
        # positive
        return "#002C0D"
    elif s <= -POS_NEG_NEUT:
        # negative:
```

```
        return "#270000"

    else:
        return app_colors['background']

def generate_table(df, max_rows=10):
    return html.Table(className="responsive-table",
                      children=[
                        html.Thead(
                          html.Tr(
                            children=[
                              html.Th(col.title()) for col in df.columns.values],
                            style={'color':app_colors['text']}
                            )
                          ),
                        html.Tbody(
                          [

                          html.Tr(
                            children=[
                              html.Td(data) for data in d
                              ], style={'color':app_colors['text'],
                                        'background-color':quick_color(d[2])}
                            )
                           for d in df.values.tolist()])
                        ]
    )


def pos_neg_neutral(col):
    if col >= POS_NEG_NEUT:
```

```
        # positive
        return 1
    elif col <= -POS_NEG_NEUT:
        # negative:
        return -1

    else:
        return 0



@app.callback(Output('recent-tweets-table', 'children'),
              [Input(component_id='sentiment_term', component_property='value')],
              events=[Event('recent-table-update', 'interval')])
def update_recent_tweets(sentiment_term):
    if sentiment_term:
        df = pd.read_sql("SELECT sentiment.* FROM sentiment_fts fts LEFT JOIN sentiment ON fts.rowid = sentiment.id WHERE fts.sentiment_fts MATCH ? ORDER BY fts.rowid DESC LIMIT 10", conn, params=(sentiment_term+'*',))
    else:
        df = pd.read_sql("SELECT * FROM sentiment ORDER BY id DESC, unix DESC LIMIT 10", conn)

    df['date'] = pd.to_datetime(df['unix'], unit='ms')

    df = df.drop(['unix','id'], axis=1)
    df = df[['date','tweet','sentiment']]

    return generate_table(df, max_rows=10)



@app.callback(Output('sentiment-pie', 'figure'),
              [Input(component_id='sentiment_term', component_property='value')],
```

```
            events=[Event('sentiment-pie-update', 'interval')])
def update_pie_chart(sentiment_term):

    # get data from cache
    for i in range(100):
        sentiment_pie_dict = cache.get('sentiment_shares', sentiment_term)
        if sentiment_pie_dict:
            break
        time.sleep(0.1)

    if not sentiment_pie_dict:
        return None

    labels = ['Positive','Negative']

    try: pos = sentiment_pie_dict[1]
    except: pos = 0

    try: neg = sentiment_pie_dict[-1]
    except: neg = 0



    values = [pos,neg]
    colors = ['#007F25', '#800000']

    trace = go.Pie(labels=labels, values=values,
                   hoverinfo='label+percent', textinfo='value',
                   textfont=dict(size=20, color=app_colors['text']),
                   marker=dict(colors=colors,
                               line=dict(color=app_colors['background'], width=2)))
```

```
    return {"data":[trace],'layout' : go.Layout(
                                          title='Positive vs Negative sentiment for "{}" (longer-term)'.format(sentiment_term),
                                          font={'color':app_colors['text']},
                                          plot_bgcolor = app_colors['background'],
                                          paper_bgcolor = app_colors['background'],
                                          showlegend=True)}



@app.callback(Output('live-graph', 'figure'),
              [Input(component_id='sentiment_term', component_property='value')],
              events=[Event('graph-update', 'interval')])
def update_graph_scatter(sentiment_term):
    try:
        if sentiment_term:
            df = pd.read_sql("SELECT sentiment.* FROM sentiment_fts fts LEFT JOIN sentiment ON fts.rowid = sentiment.id WHERE fts.sentiment_fts MATCH ? ORDER BY fts.rowid DESC LIMIT 1000", conn, params=(sentiment_term+'*',))
        else:
            df = pd.read_sql("SELECT * FROM sentiment ORDER BY id DESC, unix DESC LIMIT 1000", conn)
        df.sort_values('unix', inplace=True)
        df['date'] = pd.to_datetime(df['unix'], unit='ms')
        df.set_index('date', inplace=True)
        init_length = len(df)
        df['sentiment_smoothed'] = df['sentiment'].rolling(int(len(df)/5)).mean()
        df = df_resample_sizes(df)
        X = df.index
        Y = df.sentiment_smoothed.values
```

```
        Y2 = df.volume.values
        data = plotly.graph_objs.Scatter(
                x=X,
                y=Y,
                name='Sentiment',
                mode= 'lines',
                yaxis='y2',
                line = dict(color = (app_colors['sentiment-plot']),
                        width = 4,)
                )

        data2 = plotly.graph_objs.Bar(
                x=X,
                y=Y2,
                name='Volume',
                marker=dict(color=app_colors['volume-bar']),
                )

        return {'data': [data,data2],'layout' : go.Layout(xaxis=dict(range=[min(X),max(X)]),
                                                yaxis=dict(range=[min(Y2),max(Y2*4)],   title='Volume',
side='right'),
                                                yaxis2=dict(range=[min(Y),max(Y)],               side='left',
overlaying='y',title='sentiment'),
                                                title='Live sentiment for: "{}"'.format(sentiment_term),
                                                font={'color':app_colors['text']},
                                                plot_bgcolor = app_colors['background'],
                                                paper_bgcolor = app_colors['background'],
                                                showlegend=False)}

    except Exception as e:
        with open('errors.txt','a') as f:
```

```
            f.write(str(e))
            f.write('\n')


@app.callback(Output('historical-graph', 'figure'),
              [Input(component_id='sentiment_term', component_property='value'),
               ],
              events=[Event('historical-update', 'interval')])
def update_hist_graph_scatter(sentiment_term):
    try:
        if sentiment_term:
            df = pd.read_sql("SELECT sentiment.* FROM sentiment_fts fts LEFT JOIN sentiment ON fts.rowid = sentiment.id WHERE fts.sentiment_fts MATCH ? ORDER BY fts.rowid DESC LIMIT 10000", conn, params=(sentiment_term+'*',))
        else:
            df = pd.read_sql("SELECT * FROM sentiment ORDER BY id DESC, unix DESC LIMIT 10000", conn)
        df.sort_values('unix', inplace=True)
        df['date'] = pd.to_datetime(df['unix'], unit='ms')
        df.set_index('date', inplace=True)
        # save this to a file, then have another function that
        # updates because of this, using intervals to read the file.
        # https://community.plot.ly/t/multiple-outputs-from-single-input-with-one-callback/4970

        # store related sentiments in cache
        cache.set('related_terms', sentiment_term, related_sentiments(df, sentiment_term), 120)

        #print(related_sentiments(df,sentiment_term), sentiment_term)
        init_length = len(df)
        df['sentiment_smoothed'] = df['sentiment'].rolling(int(len(df)/5)).mean()
        df.dropna(inplace=True)
        df = df_resample_sizes(df,maxlen=500)
        X = df.index
```

```
    Y = df.sentiment_smoothed.values
    Y2 = df.volume.values

    data = plotly.graph_objs.Scatter(
        x=X,
        y=Y,
        name='Sentiment',
        mode= 'lines',
        yaxis='y2',
        line = dict(color = (app_colors['sentiment-plot']),
                width = 4,)
        )

    data2 = plotly.graph_objs.Bar(
        x=X,
        y=Y2,
        name='Volume',
        marker=dict(color=app_colors['volume-bar']),
        )

    df['sentiment_shares'] = list(map(pos_neg_neutral, df['sentiment']))

    #sentiment_shares = dict(df['sentiment_shares'].value_counts())
    cache.set('sentiment_shares', sentiment_term, dict(df['sentiment_shares'].value_counts()), 120)

    return {'data': [data,data2],'layout' : go.Layout(xaxis=dict(range=[min(X),max(X)]), # add type='category to remove gaps'
                                        yaxis=dict(range=[min(Y2),max(Y2*4)], title='Volume', side='right'),
                                        yaxis2=dict(range=[min(Y),max(Y)], side='left', overlaying='y',title='sentiment'),
```

```
                                    title='Longer-term sentiment for:
"{}"'.format(sentiment_term),
                                    font={'color':app_colors['text']},
                                    plot_bgcolor = app_colors['background'],
                                    paper_bgcolor = app_colors['background'],
                                    showlegend=False)}

    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')



max_size_change = .4

def generate_size(value, smin, smax):
    size_change = round((( (value-smin) /smax)*2) - 1,2)
    final_size = (size_change*max_size_change) + 1
    return final_size*120



# SINCE A SINGLE FUNCTION CANNOT UPDATE MULTIPLE OUTPUTS...
#https://community.plot.ly/t/multiple-outputs-from-single-input-with-one-callback/4970

@app.callback(Output('related-sentiment', 'children'),
              [Input(component_id='sentiment_term', component_property='value')],
              events=[Event('related-update', 'interval')])
def update_related_terms(sentiment_term):
```

```
    try:

        # get data from cache
        for i in range(100):
            related_terms = cache.get('related_terms', sentiment_term) # term: {mean sentiment,
count}
            if related_terms:
                break
            time.sleep(0.1)

        if not related_terms:
            return None

        buttons = [html.Button('{}({})'.format(term, related_terms[term][1]),
id='related_term_button', value=term, className='btn', type='submit', style={'background-
color':'#4CBFE1',

'margin-right':'5px',

'margin-top':'5px'}) for term in related_terms]
        #size: related_terms[term][1], sentiment related_terms[term][0]



        sizes = [related_terms[term][1] for term in related_terms]
        smin = min(sizes)
        smax = max(sizes) - smin

        buttons = [html.H5('Terms related to "{}": '.format(sentiment_term),
style={'color':app_colors['text']})]+[html.Span(term,
style={'color':sentiment_colors[round(related_terms[term][0]*2)/2],
                                                'margin-right':'15px',
                                                'margin-top':'15px',
```

```
                                        'font-
size':'{}%'.format(generate_size(related_terms[term][1], smin, smax))}) for term in
related_terms]


        return buttons


    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')


#recent-trending div
# term: [sent, size]

@app.callback(Output('recent-trending', 'children'),
              [Input(component_id='sentiment_term', component_property='value')],
              events=[Event('related-update', 'interval')])
def update_recent_trending(sentiment_term):
    try:
        query = """
                SELECT
                    value
                FROM
                    misc
                WHERE
                    key = 'trending'
        """
```

```
        c = conn.cursor()

        result = c.execute(query).fetchone()

        related_terms = pickle.loads(result[0])



## buttons = [html.Button('{}({})'.format(term, related_terms[term][1]), id='related_term_button', value=term, className='btn', type='submit', style={'background-color':'#4CBFE1',
## 'margin-right':'5px',
## 'margin-top':'5px'}) for term in related_terms]
        #size: related_terms[term][1], sentiment related_terms[term][0]


        sizes = [related_terms[term][1] for term in related_terms]
        smin = min(sizes)
        smax = max(sizes) - smin

        buttons = [html.H5('Recently Trending Terms: ', style={'color':app_colors['text']})]+[html.Span(term, style={'color':sentiment_colors[round(related_terms[term][0]*2)/2],
                                                              'margin-right':'15px',
                                                              'margin-top':'15px',
                                                              'font-size':'{}%'.format(generate_size(related_terms[term][1], smin, smax))}) for term in related_terms]
```

```
        return buttons


    except Exception as e:
        with open('errors.txt','a') as f:
            f.write(str(e))
            f.write('\n')




external_css =
["https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.min.css"]
for css in external_css:
    app.css.append_css({"external_url": css})



external_js = ['https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js',
               'https://pythonprogramming.net/static/socialsentiment/googleanalytics.js']
for js in external_js:
    app.scripts.append_script({'external_url': js})

server = app.server
dev_server = app.run_server
```