**Sentiment Analysis on Twitter Data**

**Introduction:**

Twitter is most common social media application that everyone uses. It is often the main source of information. Along with its pro come cons. One of the main disadvantage of twitter is that we have can tweet anything we want and that might be inappropriate or rude to other people. This might lead to many arguments between people. To avoid these we can calculate sentiment of the tweets a person posts. Based on the sentiment we can analyze what the user wants to convey.

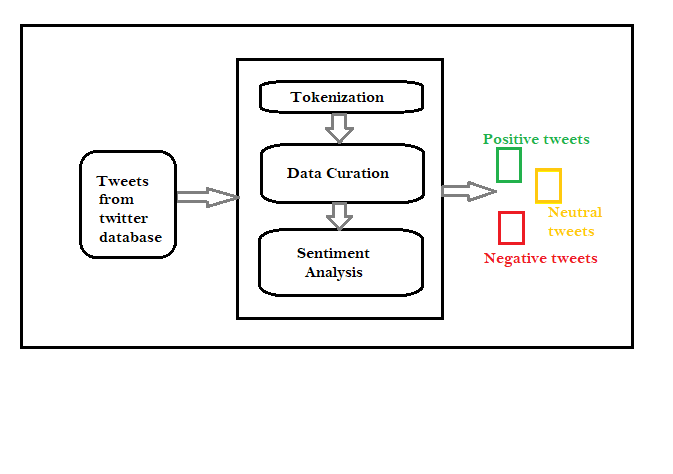


Fig 01: Sentiment analysis

**Tools used:**

For this project we have used python for programming and visualizing the results. The following are the main tools and libraries we have used in our project.

* PyCharm is an integrated development environment used specifically for the Python language.
* Tweepy library for connect twitter to our project.
* Numpy library to work with array and data frames.
* Pandas library to perform analysis on tweets
* Matplotlib library to visualize the result into a bar graph.

**Data source:**

For our project, social media platform named twitter is the main source of data. Twitter is providing a developer account that provides access to the data present in twitter data base. From twitter developer account we can access tweets by username of any person.

**About data:**

Once we connect our program to twitter developer account, we can access all the tweets from the database of twitter. We will get a twitter object which consists of following attributes.

1. Object ID: Uniquely generated number to distinguish different tweets.
2. Screen name: Username of the person who twitted.
3. Tweet: Exact text from the tweet posted.
4. Created date and time: Date and time of the tweet posted in according with UTC
5. Likes count: Number of likes a specific tweet has obtained.
6. Retweets count: Number of retweets for a specific tweets.

**Data extraction process:**

As we are obtaining data from twitter. Firstly we have to create a developer account from twitter. For this we have to visit <https://developer.twitter.com/en/apply-for-access> and enter details accordingly. After creating the developer account, we will get access tokens for connecting the twitter database to our project. We will have 4 main keys to connect developer account to python. They are ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET, CONSUMER\_KEY, and CONSUMER\_SECRET.

The following are the keys in our project.

|  |
| --- |
| # variables  ACCESS\_TOKEN = "1296122843609083908-z4qcvDA………………………."  ACCESS\_TOKEN\_SECRET = "507Q41GUsvwuGfT…………………………………."  CONSUMER\_KEY = "m0qB0dWydJljIsAPt…………………………………………"  CONSUMER\_SECRET = "nsHUyFR40fmo52A…………………………………" |

Using these token keys and tweepy library in python we can retrieve twitter objects. For this we have to authenticate by using authentication keys. The authentication pseudo code is as follows.

|  |
| --- |
| class TwitterAuthenticator():  def authenticate\_twitter\_app(self):  auth = OAuthHandler(twitter\_cred.CONSUMER\_KEY, twitter\_cred. CONSUMER\_SECRET)  auth.set\_access\_token(twitter\_cred.ACCESS\_TOKEN, twitter\_cred. ACCESS\_TOKEN\_SECRET)  return auth |

Once the authorization is done, we have to connect to twitter API and client. Using the Twitter streamer class, we can stream live data by connecting to twitter streaming API. To continue, we will have a twitter listener class which will collect all the tweets and if any error found will be sent error messages.

|  |
| --- |
| def stream\_tweets(self, fetched\_tweets\_filename, hash\_tag\_list):  Load the listener and fetch the tweets  Authenticate using twitter authentication class  def on\_data(self, data):  try:  call the method to save tweet object in data frame.  except BaseException as e:  call on error method  return false  return True  def on\_error(self, status):  print the error |

Once we get twitter object with all the required attributes, we can process the data according to the requirements. We will store the tweets along with other attributes in a data frame.

|  |
| --- |
| def tweets\_to\_data\_frame(self, tweets):  df['tweets'] gives tweet content  df['id'] gives the object ID of the tweet  df['len'] gives the length of the tweet  df['date'] gives the date and of tweet creation  df['source'] gives the source of tweet  df['likes'] gives count of likes for that tweet  df['retweets'] gives the count of retweets for this tweet.  return df |

**Data Curation:**

Once we get the twitter object our main concentration is on the content of the tweet. After getting the content we will store that content in a data frame. Before moving into any analysis we have clean tweets by removing punctuations from the tweets so that we can process the data easily. For this, we have a clean tweet method. The pseudo code is as follows.

|  |
| --- |
| class TweetAnalyzer():  def clean\_tweet(self, tweet):  return ' '.join(re.sub(“content to remove ", " content to replace with ", tweet).split()) |

After removing punctuations from the tweet we are left with actual words. These words can be categorized into two main types.

* Words that does add emotion to sentence
* Words that does not add emotion to sentence.

As we are going to evaluate and work on sentiment of tweets, we can neglect the words that doesn’t add any emotion to sentence in any circumstances. These words are also called as stop words. Below are few sample stop words.

|  |
| --- |
| stop\_words = ["I", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself",………………] |

**Analysis:**

Now we have the tweets cleaned by using the above methods. So, we can actually concentrate on the content of the tweet. Firstly we have to separate the sentence into separate words. We call this process as tokenization. Once we divide the sentence into words we can know the meaning of the word and know if the word is a positive or negative one.

We have a set of positive and negative words dictionary stored in different files name pos.txt and neg.txt. If the words in the tweet matches any of the words in positive text file, then we increment the positive word count and if the word matches any of the words in negative words file then we will increase the negative word count. When we reach end of the tweet we will compare the positive and negative word count. If positive word count is greater than negative word count then that tweet will come under positive sentiment and vice versa. But if both are equal than we will consider that tweet with neutral sentiment. The following is the pseudo code for this analysis.

|  |
| --- |
| def my\_sentiment\_analyser(self, tweet):  stop\_words = ["I", "me", "my", "myself” …]  for each word in tweet  if word not present in stop words  for each word in positive text file  if word is present in positive file increment  positive word count by 1  else if word is present in negative file then  increment positive word count by 1  if positive word count > negative word count  return 1  else if negative word count > positive word count  return -1  else  return 0 |

This method if returned 1, will represent a positive sentiment if returned -1, then a negative sentiment and if returned 0, then a neutral sentimental tweet. Then we are going to store this sentiment output in the data frame corresponding to each tweet.

**Accuracy calculation:**

To find the accuracy, we have to compare the results with a standard method of sentimental analysis. In python we have predefined libraries called TextBlob which have an analyze method that can perform sentiment analysis. The following is the pseudo code of the sentiment analysis using TextBlob.

|  |
| --- |
| def analyze\_sentiment(self, tweet):  analysis = TextBlob(tweet)  if analysis sentiment polarity score > 0:  return 1  else if analysis sentiment polarity score == 0:  return 0  else if analysis sentiment polarity score < 0:  return -1 |

So now we have results of both manually calculated sentiment and TextBlob sentiment stored in data frame. By using sklearn library we can calculate accuracy between two outputs. The following is the pseudo code for accuracy calculation.

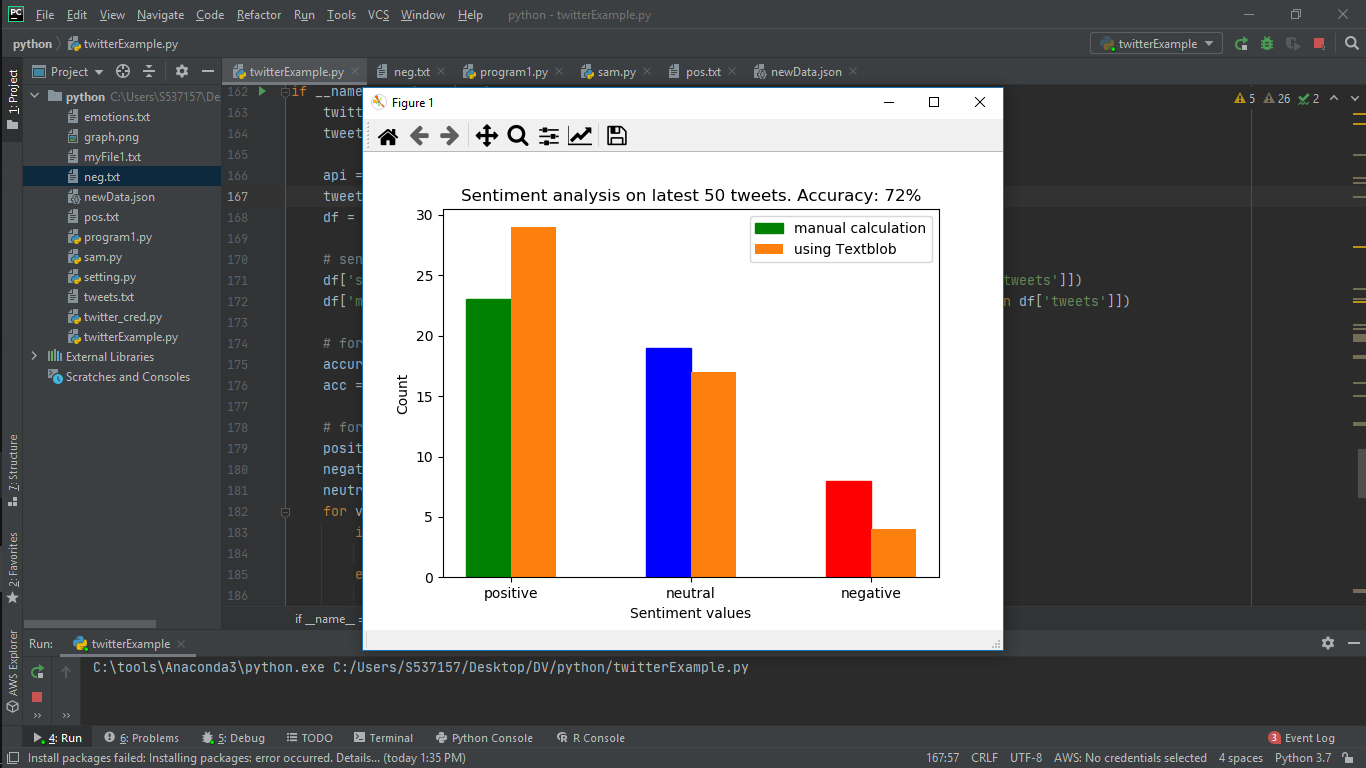
|  |
| --- |
| accuracy = accuracy score(manually calculated, sentiment using textblob) |

**Visualization:**

For visualizing this analysis, we have matplotlib library and drawn a box plot for the data. In this box plot, rectangles represent the count and x axis representing sentiment of tweets that is positive, negative or neutral. The following is the pseudo code for visualizing part.

|  |
| --- |
| My values = [manually calculated values]  Org values = [text blob values]  fig, ax = plt.subplots()  Bar1 = (index, my values….)  Bar2 = (index, org Values…) |

The output visualization is as follows.



In the above picture, we can see that out of 50 tweets collected 23 are positive tweets, 20 are neutral tweets and 7 tweets have negative words in it according to manually calculated sentiment. When coming to the sentiment calculated by text blob, 28 are positive 17 are neutral and 5 are negative sentimental tweets.

Overall accuracy is 72 % when we compare the manually calculated and using textblob library.

**Conclusion and Future work:**

To conclude, our project can separate positive, negative and neutral sentimental tweets using the content of the tweet. Output is up to 75% accurate when compared to Textblob library. The following are few things we can enhance in our project.

* Automatic execution of code to obtain latest tweet.
* Retrieving more than one person’s tweet at a time.