**SOURCE CODE:**

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Description**

**tweepy. streaming:**

a single connection is opened between your app and the API, with new results being sent through that connection whenever new matches occur

**OAuthHandler:**

We need to check the authentication of the handler handling the account hence we use OAuth to make it as painless which require to register our client application with twitter

We need required authentication credentials to be able to use the API.

Which are Consumer key, Consumer secret, Access token, Access secret

We have discussed how to generate these keys in the implementation section.

We create class to write from the stream listener and then create functions.

In the main function the stream(auth,I)

The authentation tokens,keys and the StdOutListener are considered.

**Functions:**

**1**.on\_data (self, data):

data.rstrip()-

The rstrip() method removes any trailing characters (characters at the end a string)

**2**. def on\_exception (self, exception):

print(exception)

here we print the exception

**3.** def on\_error(self, status):

print(status)

here we print the status

**IN the stream filter part of the code:**

stream.filter(track=['Trump']:The following example works fine as it returns all live tweets that include the word 'Trump’

stall\_warnings=True :Setting this parameter to the string true will cause periodic messages to be delivered if the client is in danger of being disconnected.

**get\_rules() and add\_rule:**

Here we add\_rules and get\_rules are used to add rules to the filtering of the stream here the tweets are filtered according to what is required to us and here we want the tweets with the word Trump.

Text

Description automatically generated with low confidence

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

**logging.basicConfig(level=logging.DEBUG,filename='log.txt'):**

The above command does the basic configuration for the logging system where a stream handler is created in a specific format as we have set the root logger level above .

**nlp = spacy.load('en\_core\_web\_sm'):**

Here we have to structure the unstructured data for it to make sense.Hence we use spacy library and load the en\_core\_web\_sm statmodel from the other models we have in spacy.

**def clean\_tweet(text):**

**return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ",text).split()):**

Here in the function above we clear out the data we received where a lot of words or numbers are attached to the data which is required to us so all of those are removed and the essential data is obtained

**Stop words function:**

In this function the stop words are removed.

**stop\_words = set(stopwords.words('english')):**

Here the stop words are removed which doesnt give any insight about the information which is discussed. and the stop wors in english is choosen above.

**def ratioOfTweetsContainingOpinion(text):**

Here In this function the tweets which state an opnion are filtrered with its count too.

**def ratioOfTweetsContainingTentative(text):**

Here in this function tweets with tentative words are filtered with its count too.

**def presenceOfVulgarity(text):**

Here the tweets which contain vulgarity are filtered and also counted

**def get\_tweet\_sentiment\_Pos(text):**

In this function the count of positive tweet is obtained where sentiment polarity is used to figure out that its a positive tweet.

**def get\_tweet\_sentiment\_Neu(text):**

Here the tweets which are neutral are obtained ie.., which are not positive or negative.

**def get\_tweet\_sentiment\_Neg(text):**

Here the tweets with negative polarity is counted.

**tweetsDStream = ssc.socketTextStream:**

Here Dstream is created and which have the streaming data from a tcp source.

**tweetRDD = tweetsDStream.map:**

Here the streaming data is slipt or converted as a seprate tweet.

**words = tweetRDD.map(lambda text:[ratioOfTweetsContainingTentative(text),presenceOfVulgarity(text),ratioOfTweetsContainingOpinion(text),get\_tweet\_sentiment\_Pos(text),get\_tweet\_sentiment\_Neu(text),get\_tweet\_sentiment\_Neg(text)]):**

Here the tentative,vulgarity and opnion words/tweets are slip seprately.

**countWords= words.map:**

Here the words are noted and reduced in the end to figure out the frequency of the words.

**countWords.map(lambda x (x[1],x[3],x[5],x[6],x[8],x[10],x[12])).saveAsTextFiles(path+"/text.txt"):**

Here the words are saved in a text file.

**awaitTermination:**

Here we wait for the computation to terminate

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Table

Description automatically generated

Text

Description automatically generated

**ls -lt** list the sorting by modification.grep is used to find the file text.txt and sed is use to specify multiple scripts and cut is used to split the tweets where the delimiter is given which is space and the range of fields is take as 9.

**In the next for item in $Direc loop:**

we create a file merged and for every element in files remove the ( ) with empty space using the translate command which is moved to the merged file.

**In the remaining half of the code:**

The count of each requirement is printed using the awk format which is used to specify the format in which we want the output to be printed and then the file is moved to a different location.

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated with medium confidence

Text

Description automatically generated with low confidence

**In the beginning of the code:**

In this function we read the file and split the lines

**In the first loop of the code:**

In this loop for every line in the lines if the length is more than zero then the number is appended to its respective group.

**ax1.plot(xs,ys, label="Negative")**

**ax1.set\_xlabel('time(s)'):**

Here we plot the negative on the graph as the labels are given accordingly.

**In the end part of the code:**

As we discussed the type of words are plot accordingly on the graph.And mention the location which in the above code is best which plots the graph which fits the data best.