**PROJECT BASED LEARNING LAB**

**PROJECT REPORT**

**ON**

**SENTIMENTAL ANALYSIS OF TWITTER DATA**

**SESSION:2017-18**

**Submitted By-**

**Anjali Chauhan 17303003**

**Ayushi Verma 17303010**

**Deepanshi 17303013**

**Sentimental Analysis**

Sentimental Analysis is a process of collecting and analysing the data based upon the person feelings, reviews and thoughts. Sentimental analysis is often called as opinion mining as it mines the important features from people opinions. Sentimental analysis is done by various machine learning techniques, statistical model, NLP (Natural Language Processing) for the extraction of features from a large data.

Sentimental analysis can be done at sentence, phrase or document level. In document level summary of the entire document is taken first and then it is analysed whether the sentiment is positive, negative or neutral. In phrase level, analysis pf phrases in a sentence is taken in account to check 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 another 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 messages 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 sentiments of these tweets as positive, negative or neutral with the help of different machine leaning algorithm.

Introduction to Python

Python is a high level, dynamic programming language which is used for this thesis. Python3.4 version was used as it is a mature, versatile and robust programming language. It is an interpreted language which makes the testing and debugging extremely quick as there is no compilation step. There are extensive open source libraries available for this version of python and large community of users.

**Applications of Sentimental Analysis**

Due to the increase in the sentimental analysis, social network data is on high demand. Many companies have already adopted the sentimental analysis for the process of betterment. Some of major applications are mentioned as following:

1. Word of Mouth(WOM)

Word of Mouth is the process by which the information is given from one person to another person. It would essentially help the people to take decisions. WOM has given the information about the opinions, attitudes, reactions of consumers about the related business, services and the products or even the ones that can be shared with more than one person. Here sentimental analysis comes into picture.

1. Voice of Voters

Each of the political parties usually spent a major chunk of the amount of money for the aim of campaigning for their party or for influencing the voters. Thus, if the politicians know the people opinions, reviews, suggestions, these can be done with more effect. This is how process of sentimental analysis does not only help political parties but on the other hand help the news analysts alongside.

1. Voice of the market(VOM)

Whenever a product is to be launched by a specific company the customers would to know about the product ratings, reviews and details description about it. Sentiment Analysis can help in analysing marketing, advertising and for making new strategies for promoting the product. It provides the customer an opportunity to choose the best among them all.

1. Brand Reputation Management(BRM)

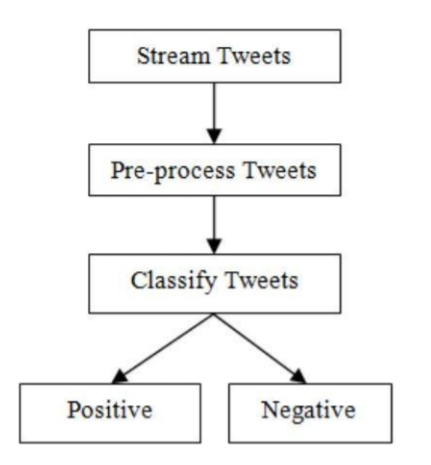
Sentiment analysis would help to determine how would be a company’s brand, service and the service or product that would be perceived by the online community. Brand Reputation Management will be concerned about the management of the reputation of market. It has focuses on the company and product rather than customer. Thus the opportunities were created for the purpose of managing and strengthening the brand reputation of the organisation.

**IMPLEMENTATION**

Data collection is not a simple task, as it may seem. Various decisions have to be made for collecting data. For our project we maintain dataset for training, testing and made for collecting data. Under implementation we are going to study how data is collected, stored, processed and classified. Before discussing these process and different dataset, let us discuss our proposed architecture.

Proposed Architecture

As our goal is to achieve sentiment analysis for data provided from twitter. We are going to build classifier which consists of different machine learning classifiers. After the classifier is ready, follow the following steps:



Step -1 First we are going to stream tweets in our build classifier with help of Tweepy library in python.

Step-2 Then we pre-process these tweets, so that they can be fir for mining and feature extraction.

Step-3 After pre-processing we pass this data in our trained classifier, which then classify them into positive, negative or neutral.

**REQUIREMENTS**

First of all, we need to have [Python](https://www.python.org/downloads/) installed.   
Install [Anaconda](https://www.anaconda.com/download/), which is a very useful Python distribution to manage packages that includes a lot of useful tools, such as [Jupyter Notebooks](http://jupyter.org/).

The required libraries are:

* [NumPy](http://www.numpy.org/): This is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.
* [Pandas](http://pandas.pydata.org/): This is an open source library providing high-performance, easy-to-use data structures and data analysis tools.
* [Tweepy](http://www.tweepy.org/): This is an easy-to-use Python library for accessing the Twitter API.
* [Matplotlib](http://matplotlib.org/): This is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
* [Seaborn](https://seaborn.pydata.org/): This is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.
* [Textblob](https://textblob.readthedocs.io/): This is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks.

**Working**

* Extracting twitter data using tweepy and panda library

Firstly, import all the required libraries.

*import tweepy # To consume Twitter's API*

*import pandas as pd # To handle data*

*import numpy as np # For number computing*

*import csv*

*# For plotting and visualization:*

*from IPython.display import display*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*%matplotlib inline*

* Create a twitter app.

In order to extract tweets for a posterior analysis, we need to access to our Twitter account and create an app.

Code snippet-

*CONSUMER\_KEY = '7H0tnWTrwJUaWBGY5RCub8LEW'*

*CONSUMER\_SECRET='HxuA9wyMy48WqSrFYcdeB9fMfibPvPuMVrKSILDQ2q9tR0dsRY'*

*ACCESS\_TOKEN='902769774022840320-Tg7ZYxxDVrAwKno9ctuMie8bNxv5nRH'*

*ACCESS\_SECRET='a8DNJVG9B2y2G7B6fxUPcl4zzbLjsalDDhP2DAGgyRlV1'*

we will create a function to allow us our keys authentication. We will add this function in another cell of code and we will run it:

Code Snippet-

*# We import our access keys:*

*from credentials import \**

*# This will allow us to use the keys as variables*

*# API's setup:*

*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*

* Tweets Extraction

Now, we've created a function to setup the Twitter API, we can use this function to create an "extractor" object. After this, we will use Tweepy's function extractor.user\_timeline(screen\_name, count) to extract from screen\_name's user the quantity of count tweets.

Code snippet-

*# We create an extractor object:*

*extractor = twitter\_setup()*

*screen\_name="ICC"*

*# We create a tweet list as follows:*

*tweets = extractor.user\_timeline(screen\_name, count=200)*

*print("Number of tweets extracted: {}.\n".format(len(tweets)))*

*# We print the most recent 5 tweets:*

*print("100 recent tweets:\n")*

*for tweet in tweets[:100]:*

*print(tweet.text)*

*print()*

* Create a csv file and store the tweets details in it.

*#Create a CSV file and store the tweet details in it.*

*output=[[tweet.id,tweet.created\_at,tweet.text.encode("utf8"),len(tweet.text),tweet.source,tweet.favorite\_count,tweet.retweet\_count] for tweet in tweets]*

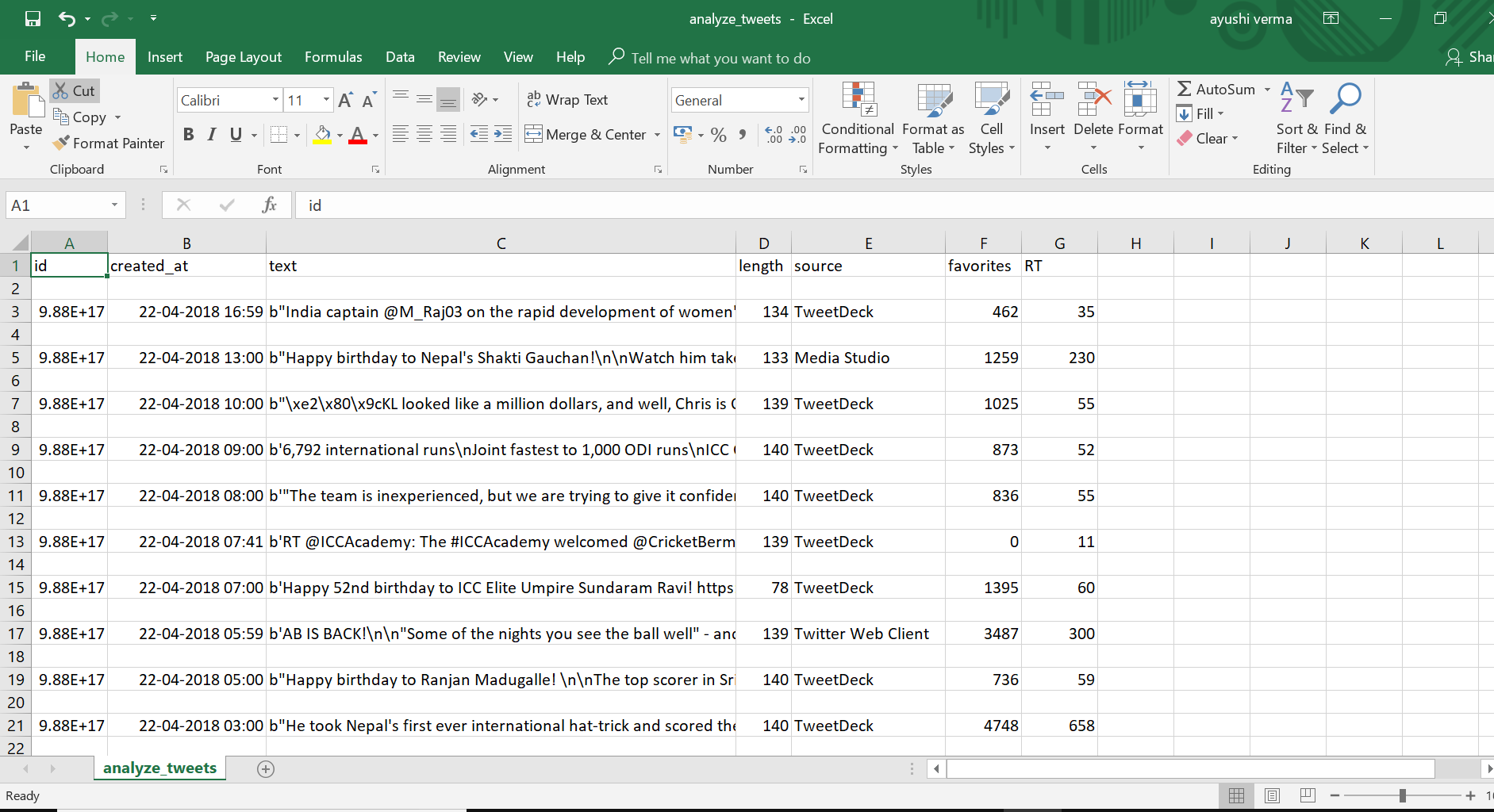
*with open(r'C:\Users\Ayushi\Desktop\analyze\_tweets.csv','w') as file:*

*writer = csv.writer(file)*

*writer.writerow(["id","created\_at","text","length","source","favorites","RT"])*

*writer.writerows(output)*

output-

**

### Creating a (pandas) DataFrame

We now have initial information to construct a pandas DataFrame, in order to manipulate the info in a very easy way.

So, using Python's list comprehension:

Code snippet-

data = pd.DataFrame(data=[tweet.text for tweet in tweets], columns=['Tweets'])

### Adding relevant info to our dataframe

We can obtain a lot of data from a single tweet. But not all this data is always useful for specific stuff. Here, we well just add some data to our dataframe. For this we will use Pythons list comprehension and a new column will be added to the dataframe by just simply adding the name of the content between square brackets and assign the content. The code goes as...:

*# We add relevant data:*

*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])*

# Visualization and basic statistics

### Averages and popularity

Here, first calculate some basic statistical data, such as the mean of the length of characters of all tweets, the tweet with more likes and retweets, etc.

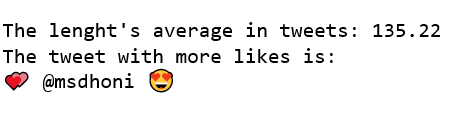
To obtain the mean, using numpy:

*# We extract the mean of lengths:*

*mean = np.mean(data['len'])*

*print("The length's average in tweets: {}".format(mean))*

*OUTPUT-*

**

*# We extract the tweet with more FAVs and more RTs:*

*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]))*

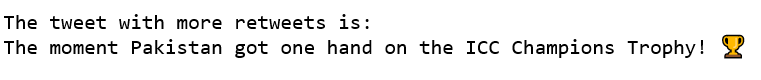
*OUTPUT-*

**

*# Max RTs:*

*print("The tweet with more retweets is: \n{}".format(data['Tweets'][rt]))*

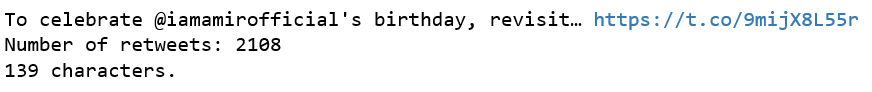
*OUTPUT-*

**

*print("Number of retweets: {}".format(rt\_max))*

*print("{}characters.\n".format(data['len'][rt]))*

*OUTPUT-*

**

### Time series

Pandas has its own object for time series. Since we have a whole vector with creation dates, we can construct time series respect tweets lengths, likes and retweets.

Code snippet-

*tlen = pd.Series(data=data['len'].values, index=data['Date'])*

*tfav = pd.Series(data=data['Likes'].values, index=data['Date'])*

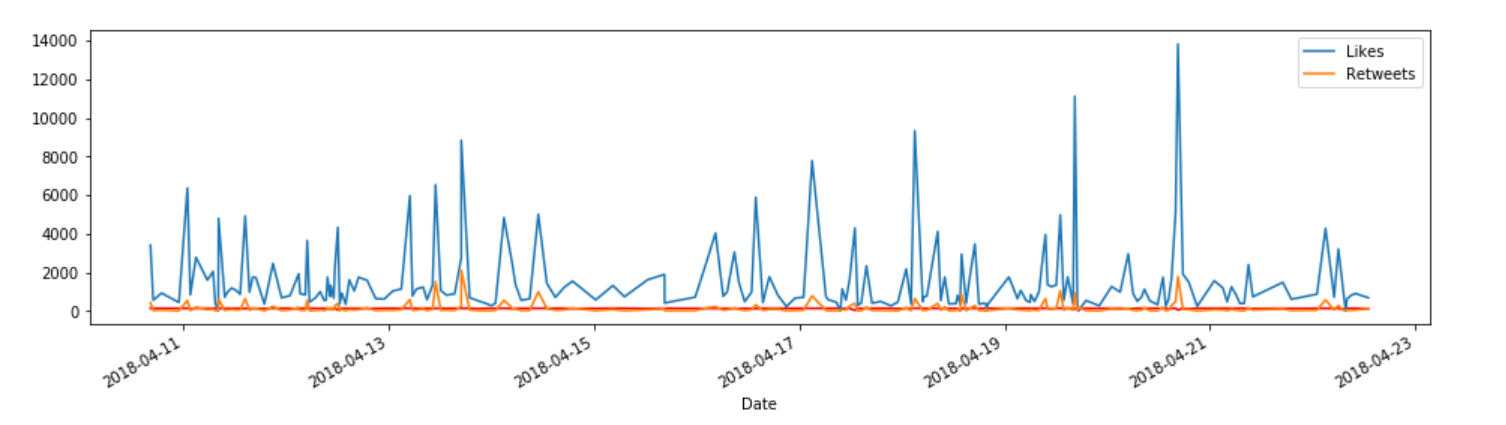
*tret = pd.Series(data=data['RTs'].values, index=data['Date'])*

And if we want to plot the time series, pandas already has its own method in the object. We can plot a time series as follows:

*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);*

**

### Pie charts of sources

Now we will plot the sources in a pie chart, since we realized that not every tweet is tweeted from the same source. We first clean all the sources:

*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))*

*# We create a numpy vector mapped to labels:*

*percent = np.zeros(len(sources))*

*for source in data['Source']:*

*for index in range(len(sources)):*

*if source == sources[index]:*

*percent[index] += 1*

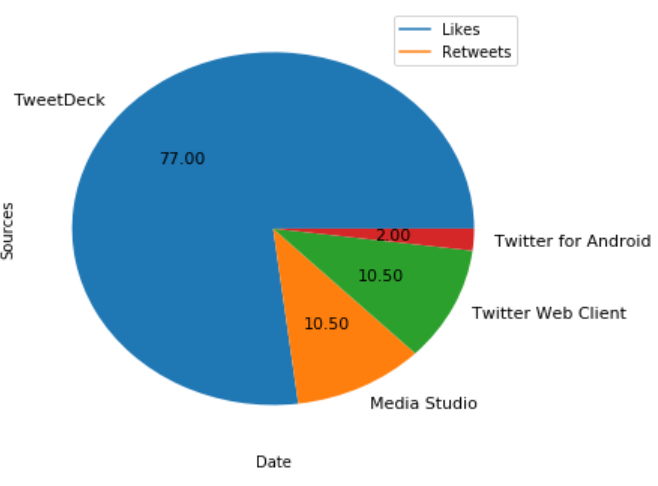
*pass*

*percent /= 100*

*pie\_chart = pd.Series(percent, index=sources, name='Sources')*

*pie\_chart.plot.pie(fontsize=11, autopct='%.2f', figsize=(6, 6));*

output*-*

**

# Sentiment analysis

### Importing textblob

Textblob will allow us to do sentiment analysis in a very simple way. We will also use the re library from Python, which is used to work with regular expressions. Here, two utility functions are used:

a) clean text (which means that any symbol distinct to an alphanumeric value will be remapped into a new one that satisfies this condition)

b) create a classifier to analyse the polarity of each tweet after cleaning the text in it.

Code Snippet-

*from textblob import TextBlob*

*import re*

*def clean\_tweet(tweet):*

*'''*

*Utility function to clean the text in a tweet by removing*

*links and special characters using regex.*

*'''*

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

*def analize\_sentiment(tweet):*

*'''Utility function to classify the polarity of a tweet using textblob. '''*

*analysis = TextBlob(clean\_tweet(tweet))*

*if analysis.sentiment.polarity > 0:*

*return 1*

*elif analysis.sentiment.polarity == 0:*

*return 0*

*else:*

*return -1*

The way it works is that textblob already provides a trained analyser. Textblob can work with different machine learning models used in natural language processing.

This added column will contain the sentiment analysis and we can plot the data-frame to see the update:

*# We create a column with the result of the analysis:*

*data['SA'] = np.array([ analize\_sentiment(tweet) for tweet in data['Tweets']])*

*# We display the updated dataframe with the new column:*

*display(data.head(100))*

output-

**

### Analysing the results

To have a simple way to verify the results, we will count the number of neutral, positive and negative tweets and extract the percentages.

*# We construct lists with classified tweets:*

*pos\_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] > 0]*

*neu\_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] == 0]*

*neg\_tweets = [ tweet for index, tweet in enumerate(data['Tweets']) if data['SA'][index] < 0]*

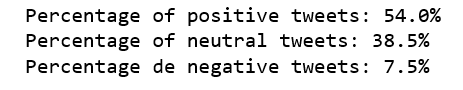
*# We print percentages:*

*print("Percentage of positive tweets: {}%".format(len(pos\_tweets)\*100/len(data['Tweets'])))*

*print("Percentage of neutral tweets: {}%".format(len(neu\_tweets)\*100/len(data['Tweets'])))*

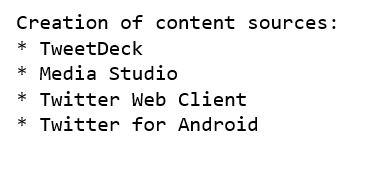
*print("Percentage de negative tweets: {}%".format(len(neg\_tweets)\*100/len(data['Tweets'])))*

*OUPUT-*

**

An interesting thing is to analyse the polarity of the tweets from different sources, it might be deterministic that by only considering the tweets from one source the polarity would result more positive/negative.

Output of content sources-



CONCLUSION

Sentimental Analysis is used to identify people’s opinion, attitude, and emotional states. The view 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 extract the twitter posts from twitter by users. The system can also compute the frequency of each term in tweet.

As we saw, we can extract, manipulate, visualize and analyse data in a very simple way with Python.