CHAPTER 1

Introduction

1.1 Problem Defination

Sentiment analysis in the domain of micro-blogging is a relatively new research topic. so there is still a lot of room for further research in this area. Decent amount of related prior work has been done on sentiment analysis of user reviews, documents, web blogs/articles and general phrase level sentiment analysis. These differ from twitter mainly because of the limit of 140 characters per tweet. which forces the user to express opinion compressed in very short text. The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes, but the manual labeling required for the supervised approach is very expensive. Some work has been done on unsupervised and semi-supervised approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need of proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications. The problem in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level. Where the expressed opinion is a document, or a sentence is positive, negative and neutral.

1.2 Aim and Objective :

- To implement an algorithm for automatic classification of text into positive, negative and neutral.
- Sentiment Analysis to determine the attitude of the mass is positive, negative and neutral towards the subject of interest.

1.3 Sentimental analysis

Sentiment analysis refers to the broad area of natural language processing, which deals with the computational study of opinions, sentiments and emotions expressed in text.

Sentiment Analysis (SA) or Opinion Mining (OM) aims at learning people's opinions, attitudes and emotions towards an entity. The entity can represent individuals, events or topics. An immense amount of research has been performed in the area of sentiment analysis. But most of them focused on classifying formal and larger pieces of text data like reviews.

With the wide popularity of social networking and micro blogging websites and an immense amount of data available from these resources, research projects on sentiment analysis have witnessed a gradual domain shift. The past few years have witnessed a huge growth in the use of micro blogging platforms. Popular micro blogging websites like Twitter have evolved to become a source of varied information. This diversity in the information owes to such micro blogs being elevated as platforms where people post real time messages about their opinions on a wide variety of topics, discuss current affairs and share their experience on products and services they use in daily life.

CHAPTER 2

INTRODUCTION TO TECHNOLOGY USED

2.1 PYTHON

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.
Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

2.2 Naive Bayes Classification:

Many language processing tasks are tasks of classification, although luckily our classes are much easier to define than those of Borges. In this classification we present the naive Bayes algorithms classification, demonstrated on an important classification problem: text categorization, the task of classifying an entire text by assigning it a text categorization label drawn from some set of labels.

We focus on one common text categorization task, sentiment analysis, the ex-sentiment analysis traction of sentiment, the positive or negative orientation that a writer expresses toward some object. Are view of a movie, book, or product on the web expresses the author's sentiment toward the product, while an editorial or political text expresses sentiment toward a candidate or political action. Automatically extracting consumer sentiment is important for marketing of any sort of product, while measuring public sentiment is important for politics and also for market prediction. The simplest version of sentiment analysis is a binary classification task, and the words of there view provide excellent cues. Consider, for example, the following phrases extracted from positive and negative reviews of movies and restaurants,. Words like great, richly, awesome, and pathetic, and awful and ridiculously are very informative cues:

+ ...zany characters and richly applied satire, and some great plot twists

− It was pathetic. The worst part about it was the boxing scenes...

+ ...awesome caramel sauce and sweet toasty almonds. I love this place!

− ...awful pizza and ridiculously overpriced...

Naive Bayes is a probabilistic classifier, meaning that for a document d, out of all classes c∈C the classifier returns the class ˆ c which has the maximum posterior probability given the document. In Eq. 1 we use the hat notation to mean "our estimate of the correct class".

c = argmax P(c|d) where c∈C

This idea of Bayesian inference has been known since the work of Bayes (1763), Bayesian inference and was first applied to text classification by Mosteller and Wallace (1964).

The intuition of Bayesian classification is to use Bayes' rule to transform Eq.6.1intoother probabilities that have some useful properties. Bayes' rule is presented in Eq. 2; it gives us a way to break down any conditional probability P(x|y) into three other probabilities:

P(x|y) =P(y|x)P(x) /P(y)

• Many language processing tasks can be viewed as tasks of classification. learn to model the class given the observation.

• Text categorization, in which an entire text is assigned a class from a finite set, comprises such tasks as sentiment analysis, spam detection, email classification, and authorship attribution.

• Sentiment analysis classifies a text as reflecting the positive ,negative and neutral orientation (sentiment) that a writer expresses toward some object.

• Naive Bayes is a generative model that make the bag of words assumption

(positiondoesn’tmatter)and the conditional independence assumption(words are conditionally independent of each other given the class)

• Naive Bayes with binarized features seems to work better for many text classification tasks.

The TextBlob package for Python is a convenient way to do a lot of Natural Language Processing (NLP) tasks. For example:

From textblob import TextBlob

TextBlob(“not a very great calculation”).sentiment

This tells us that the English phrase “not a very great calculation” has a polarity of about -0.3, meaning it is slightly negative, and a subjectivity of about 0.6, meaning it is fairly subjective.

There are helpful comments like this one, which gives us more information about the numbers we're interested in:

Each word in the lexicon has scores for:
1) polarity: negative vs. positive (-1.0 => +1.0)
2) subjectivity: objective vs. subjective (+0.0 => +1.0) # 3) intensity: modifies next word? (x0.5 => x2.0)

Word	Polarity	Subjectivity	Intensity
Great	1.0	1.0	10
Great	1.0	1.0	1.0
Great	0.4	0.2	1.0
Great	0.8	0.8	1.0

When calculating sentiment for a single word, TextBlob uses a sophisticated technique known to Mathematicians as “averaging”.
TextBlob(“great").sentiment

Sentiment(polarity=0.8, subjectivity=0.75)

TextBlob("not great").sentiment

Sentiment(polarity=-0.4, subjectivity=0.75)

Negation multiplies the polarity by -0.5, and doesn't affect subjectivity.

TextBlob also handles modifier words! Here's the summarized record for “very” from the lexicon:

Word	Polarity	Subjectivity	Intensity
Very	0.2	0.3	1.3

Recognizing “very” as a modifier word, TextBlob will ignore polarity and subjectivity and just use intensity to modify the following word:

TextBlob("very great").sentiment

Sentiment(polarity=1.0, subjectivity=0.9750000000000001)

The polarity gets maxed out at 1.0, but you can see that subjectivity is also modified by “very” to become 0.75·1.3=0.9750.75·1.3=0.975.

Negation combines with modifiers in an interesting way: in addition to multiplying by -0.5 for the polarity, the inverse intensity of the modifier enters for both polarity and subjectivity.

TextBlob("not very great").sentiment

#Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769)
polarity=−0.5·11.3·0.8≈−0.31polarity=−0.5·11.3·0.8≈−0.31
subjectivity=11.3·0.75≈0.58subjectivity=11.3·0.75≈0.58
TextBlob will ignore one-letter words in its sentiment phrases, which means things like this will work just the same way:

TextBlob("not a very great").sentiment

##Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769) And TextBlob will ignore words it doesn't know anything about:
TextBlob("not a very great calculation").sentiment

##Sentiment(polarity=-0.3076923076923077, subjectivity=0.5769230769230769)

TextBlob goes along finding words and phrases it can assign polarity and subjectivity to, and it averages them all together for longer text.

CHAPTER 3

WORKING

Installation:

- **Tweepy:** tweepy is the python client for the official Twitter API. Install it using following pip command:

  ```
  pip install tweepy
  ```

- **TextBlob:** textblob is the python library for processing textual data. Install it using following pip command:

  ```
  pip install textblob
  ```

  Also, we need to install some NLTK corpora using following command:

  ```
  python -m textblob.download_corpora
  ```

  (Corpora is nothing but a large and structured set of texts.)

Authentication:

In order to fetch tweets through Twitter API, one needs to register an App through their twitter account. Follow these steps for the same:

- Open this link and click the button: 'Create New App'
- Fill the application details. You can leave the callback url field empty.
- Once the app is created, you will be redirected to the app page.
- Open the 'Keys and Access Tokens' tab.
- Copy 'Consumer Key', 'Consumer Secret', 'Access token' and 'Access Token Secret'.

We follow these 3 major steps in our program:

- Authorize twitter API client.
- Make a GET request to Twitter API to fetch tweets for a particular query.
- Parse the tweets. Classify each tweet as positive, negative or neutral.

CHAPTER 4

REQUIREMENTS

Hardward requirements :

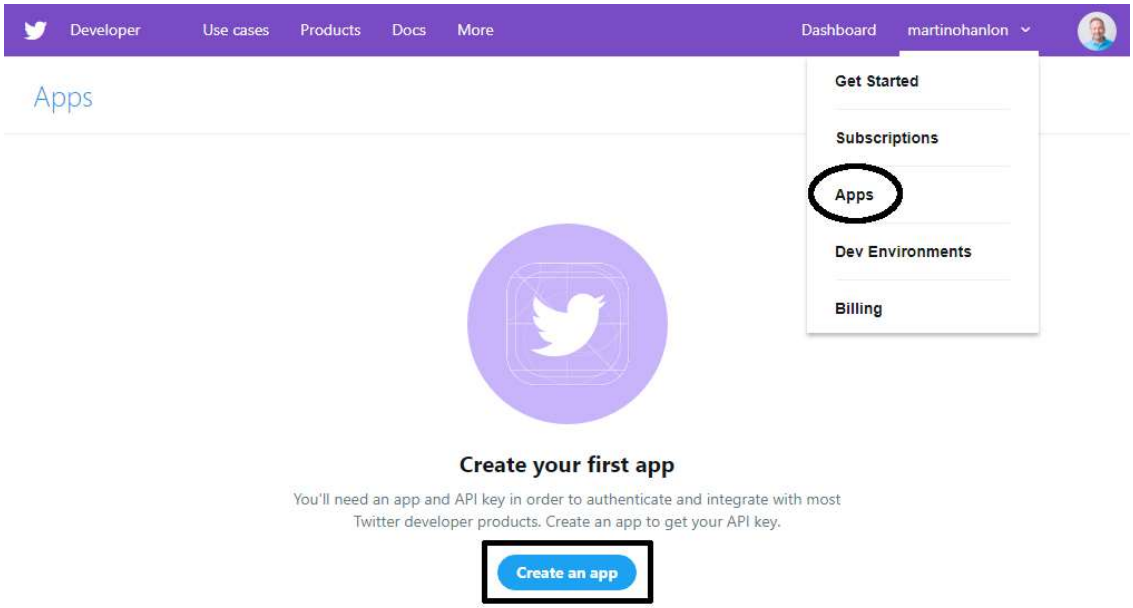
- Laptop
- Processor with 8 GB RAM

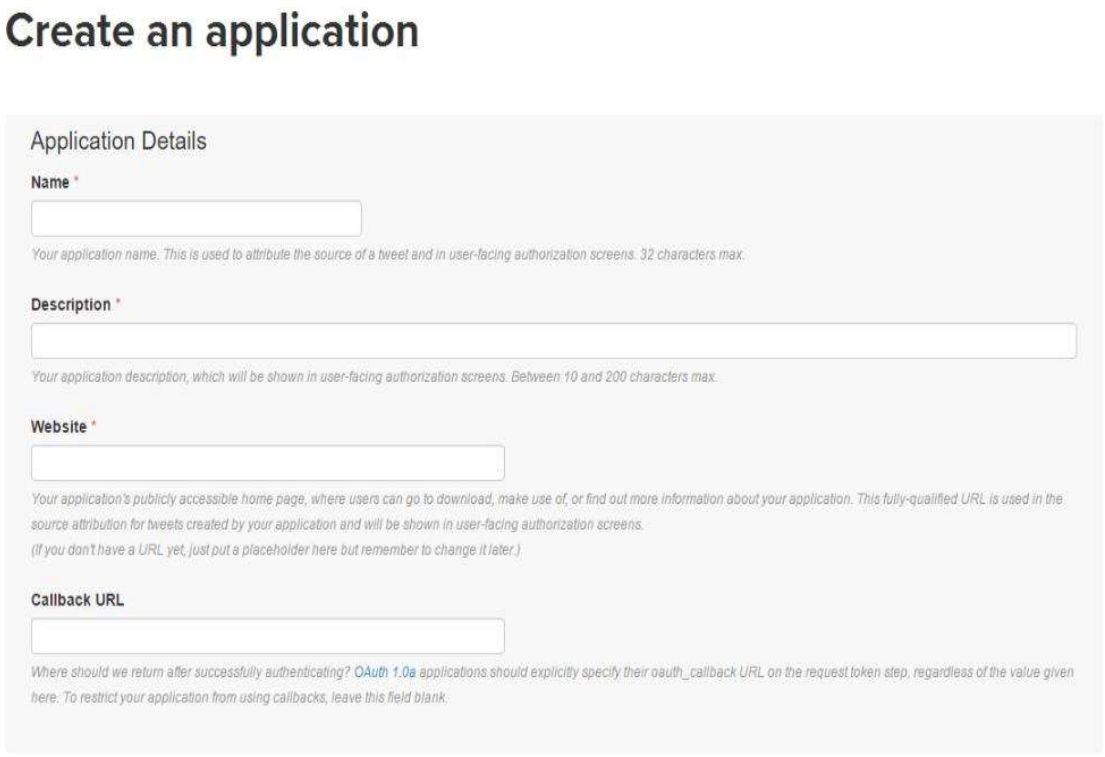
Software requirements:

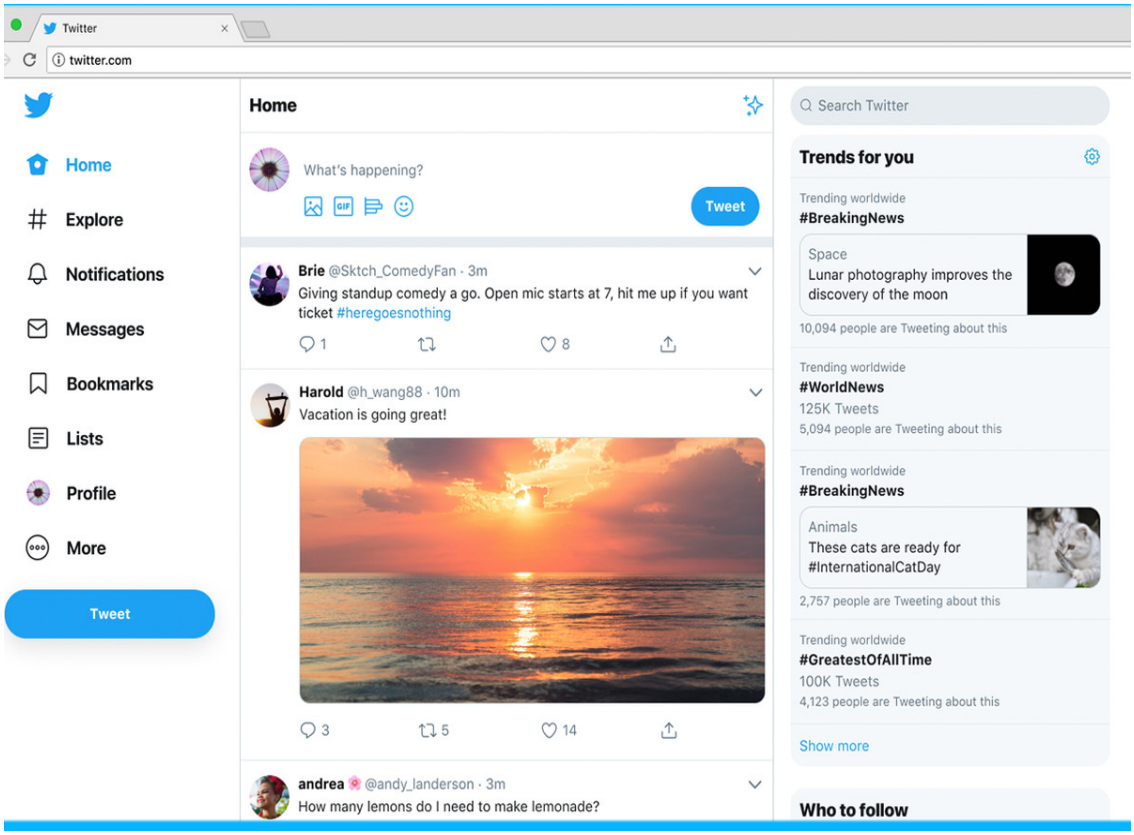
- Operating system- Windows 10
- Python IDE- Colab
- Browser- Chrome

CHAPTER 5

SNAPSHOT

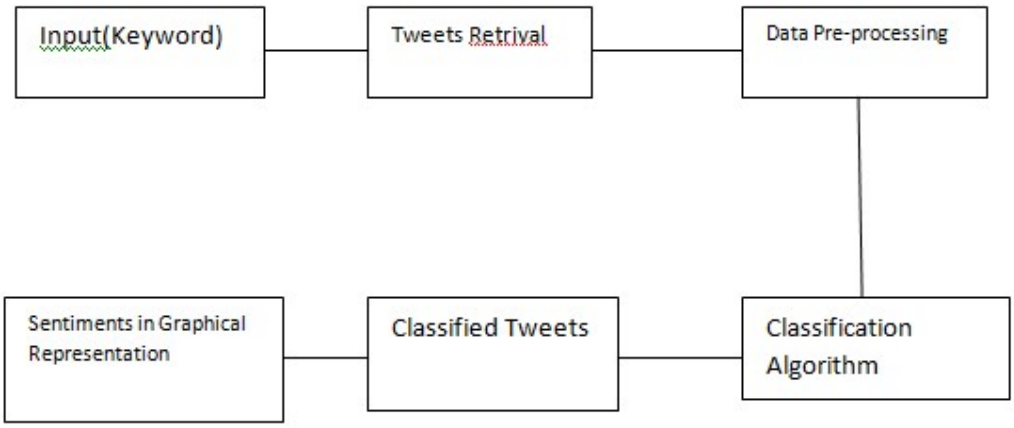





Twitter
twitter.com
Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet
Home
What's happening?
Tweet
Brie @Sktch_ComedyFan · 3m
Giving standup comedy a go. Open mic starts at 7, hit me up if you want ticket #heregoesnothing
1
8
Harold @h_wang88 · 10m
Vacation is going great!
3
5
14
andrea @andy_landerson · 3m
How many lemons do I need to make lemonade?
Search Twitter
Trends for you
Trending worldwide
#BreakingNews
Space
Lunar photography improves the discovery of the moon
10,094 people are Tweeting about this
Trending worldwide
#WorldNews
125K Tweets
5,094 people are Tweeting about this
Trending worldwide
#BreakingNews
Animals
These cats are ready for #InternationalCatDay
2,757 people are Tweeting about this
Trending worldwide
#GreatestOfAllTime
100K Tweets
4,123 people are Tweeting about this
Show more
Who to follow

CHAPTER 6

DESIGN



Input(KeyWord):

Data in the form of raw tweets is acquired by using the Python library "tweepy" which provides a package for simple twitter streaming API . This API allows two modes of accessing tweets: SampleStream and FilterStream. SampleStream simply delivers a small, random sample of all the tweets streaming at a real time. FilterStream delivers tweet which match a certain criteria. It can filter the delivered tweets according to three criteria:

- Specific keyword to track/search for in the tweets
- Specific Twitter user according to their name
- Tweets originating from specific location(s) (only for geo-tagged tweets).

A programmer can specify any single one of these filtering criteria or a multiple combination of these. But for our purpose we have no such restriction and will thus stick to the SampleStream mode.

Since we wanted to increase the generality of our data, we acquired it in portions at different points of time instead of acquiring all of it at one go. If we used the latter approach then the generality of the tweets might have been compromised since a significant portion of the tweets would be referring to some certain trending topic and would thus have more or less of the same general mood or sentiment. This

phenomenon has been observed when we were going through our sample of acquired tweets. For example the sample acquired near Christmas and New Year's had a significant portion of tweets referring to these joyous events and were thus of a generally positive sentiment. Sampling our data in portions at different points in time would thus try to minimize this problem. Thus forth, we acquired data at four different points which would be 17th of December 2015, 29th of December 2015, 19th of January 2016 and 8th of February 2016.

A tweet acquired by this method has a lot of raw information in it which we may or may not find useful for our particular application. It comes in the form of the python "dictionary" data type with various key-value pairs. A list of some key-value pairs are given below:

- Whether a tweet has been favourited
- User ID
- Screen name of the user
- Original Text of the tweet
- Presence of hashtags
- Whether it is a re-tweet
- Language under which the twitter user has registered their account
- Geo-tag location of the tweet
- Date and time when the tweet was created

Since this is a lot of information we only filter out the information that we need and discard the rest. For our particular application we iterate through all the tweets in our sample and save the actual text content of the tweets in a separate file given that language of the twitter is user's account is specified to be English. The original text content of the tweet is given under the dictionary key "text" and the language of user's account is given under "lang".

Tweets Retrieval:

Since human labelling is an expensive process we further filter out the tweets to be labelled so that we have the greatest amount of variation in tweets without the loss of generality. The filtering criteria applied are stated below:

• Remove Retweets (any tweet which contains the string “RT”)

• Remove very short tweets (tweet with length less than 20 characters)

• Remove non-English tweets (by comparing the words of the tweets with a list of 2,000 common English words, tweets with less than 15% of content matching threshold are discarded)

• Remove similar tweets (by comparing every tweet with every other tweet, tweets with more than 90% of content matching with some other tweet is discarded)

After this filtering roughly 30% of tweets remain for human labelling on average per sample, which made a total of 10,173 tweets to be labelled.

Data Preprocessing:

Data preprocessing consists of three steps:
1) tokenization,
2) normalization, and
3) part-of-speech (POS) tagging.

Tokenization:

It is the process of breaking a stream of text up into words, symbols and other meaningful elements called “tokens”. Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that we can look at tokens as individual components that make up a tweet.Emoticons and abbreviations (e.g., *OMG, WTF, BRB*) are identified as part of the tokenization process and treated as individual tokens.

Normalization:

For the normalization process, the presence of abbreviations within a tweet is noted and then abbreviations are replaced by their actual meaning (e.g., *BRB − > be right back*). We also identify informal intensifiers such as all-caps (e.g., *I LOVE this show!!!* and character repetitions (e.g., *I’ve got a mortgage!! happyyyyyy”*), note their presence in the tweet. All-caps words are made into lower case, and instances of repeated characters are replaced by a single character. Finally, the

presence of any special Twitter tokens is noted (e.g., #hashtags, usertags, and URLs) and placeholders indicating the token type are substituted. Our hope is that this normalization improves the performance of the POS tagger, which is the last preprocessing step.

Part-of-speech:

POS-Tagging is the process of assigning a tag to each word in the sentence as to which grammatical part of speech that word belongs to, i.e. noun, verb, adjective, adverb, coordinating conjunction etc. For each tweet, we have features for counts of the number of verbs, adverbs, adjectives, nouns, and any other parts of speech.

Classification Algorithem:

Let's build a sentiment analysis of Twitter data to show how you might integrate an algorithm like this into your applications. We'll first start by choosing a topic, then we

will gather tweets with that keyword and perform sentiment analyis on those tweets. We'll end up with an overall impression of whether people view the topic positively or not.

Step 1: Gather Tweets

First, choose a topic you wish to analyze. Inside sentiment-analysis.js, you can define input to be whatever phrase you like. In this example, we'll use a word we expect to return positive results.

Step 2: Perform Sentiment Analysis on Tweets

After gathering and cleaning our data set, we are ready to execute the sentiment analysis algorithm on each tweet. Then, we will calculate an average score for all the tweets combined.

Classified Tweets:

We labelled the tweets in three classes according to sentiments expressed/observed in the tweets: positive, negative and neutral We gave the following guidelines to our labellers to help them in the labelling process

Positive:

If the entire tweet has a positive/happy/excited/joyful attitude or if something is mentioned with positive connotations. Also if more than one sentiment is expressed in the tweet but the positive sentiment is more dominant. Example: “4 more years of being in shithole Australia then I move to the USA! :D”.

Negative:

If the entire tweet has a negative/sad/displeased attitude or if something is mentioned with negative connotations. Also if more than one sentiment is expressed in the tweet but the negative sentiment is more dominant. Example: “I want an android now this iPhone is boring :S”.

Neutral:

If the creator of tweet expresses no personal sentiment/opinion in the tweet and merely transmits information. Advertisements of different products would be labelled under this category. Example: “US House Speaker vows to stop Obama contraceptive rule... http://t.co/cyEWqKlE”.

CHAPTER 7

CODE

```
import re
import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob

class TwitterClient(object):
'''
Generic Twitter Class for sentiment analysis.
'''
def __init__(self):
'''
Class constructor or initialization method.
'''
# keys and tokens from the Twitter Dev Console
consumer_key = 'pVj637AtLJLwDrMNOc0Mp9OQA'
consumer_secret = 'uSLcXaG7DvS1waqM6R49CWL5uYdqhivGgFk7cWgqGocKyam
9Pn'
access_token = '712620235694559232-79lS0bg1DskepVpJlTj5QrUkm831R3s'
access_token_secret = 'I7K50bEgAcKNSCLcxTo5WybRCvwksFy3KkUQqQyzLJiZo'

# attempt authentication
try:
# create OAuthHandler object
self.auth = OAuthHandler(consumer_key, consumer_secret)
# set access token and secret
self.auth.set_access_token(access_token, access_token_secret)
# create tweepy API object to fetch tweets
self.api = tweepy.API(self.auth)
```

```
except:
print("Error: Authentication Failed")

def clean_tweet(self, tweet):
'''
Utility function to clean tweet text by removing links, special characters
using simple regex statements.
'''
return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-
z \t])|(\w+:\/\/\S+)", " ", tweet).split())

def get_tweet_sentiment(self, tweet):
'''
Utility function to classify sentiment of passed tweet
using textblob's sentiment method
'''
# create TextBlob object of passed tweet text
analysis = TextBlob(self.clean_tweet(tweet))
# set sentiment
if analysis.sentiment.polarity >0:
return 'positive'
elif analysis.sentiment.polarity == 0:
return 'neutral'
else:
return 'negative'

def get_tweets(self, query, count = 10):
'''
Main function to fetch tweets and parse them.
'''
# empty list to store parsed tweets
tweets = []
```

```
try:
# call twitter api to fetch tweets
fetched_tweets = self.api.search(q = query, count = count)

# parsing tweets one by one
for tweet in fetched_tweets:
# empty dictionary to store required params of a tweet
parsed_tweet = {}

# saving text of tweet
parsed_tweet['text'] = tweet.text
# saving sentiment of tweet
parsed_tweet['sentiment'] = self.get_tweet_sentiment(tweet.text)

# appending parsed tweet to tweets list
if tweet.retweet_count > 0:
# if tweet has retweets, ensure that it is appended only once
if parsed_tweet not in tweets:
tweets.append(parsed_tweet)
else:
tweets.append(parsed_tweet)

# return parsed tweets
return tweets

except tweepy.TweepError as e:
# print error (if any)
print("Error : " + str(e))

def main():
# creating object of TwitterClient Class
api = TwitterClient()
# calling function to get tweets
```

```
tweets = api.get_tweets(query = 'Narendra Modi', count = 200)
# picking positive tweets from tweets
ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']
# percentage of positive tweets
print("Positive tweets percentage: {} %".format(100*len(ptweets)/len(tweets)))
# picking negative tweets from tweets
ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']
# percentage of negative tweets
print("Negative tweets percentage: {} %".format(100*len(ntweets)/len(tweets)))

# percentage of neutral tweets
print("Neutral tweets percentage: {} %".format(100*(len(tweets) - len(ntweets) -
 len(ptweets))/len(tweets)))
#print("hello")
# printing first 5 positive tweets
print("\n\nPositive tweets:")
for tweet in ptweets[:10]:
print(tweet['text'])

# printing first 5 negative tweets
print("\n\nNegative tweets:")
for tweet in ntweets[:10]:
print(tweet['text'])

if __name__ == "__main__":
# calling main function
main()
```

output:

`

Positive tweets percentage: 32.5 %
Negative tweets percentage: 15.0 %
Neutral tweets percentage: 52.5 %

Positive tweets:
RT @pedro_pascal_: @hareemazeem23 it's a terrorist organisation in a form of political party...just have a look at their top leaders
Lk Adw…
RT @geetv79: - Anonymous donors were allowed to donate expired electoral bonds worth Rs 10 Crore.
- SBI was made to accept the expired bon…
RT @geetv79: PMO illegally directed FM to break its own rules on sale of electoral bonds by opening “Special” 10-day window before Karnatak…
@nit_set Finance Ministry told SBI to accept expired electoral bonds worth Rs 10 crore as a one time exception. The… https://t.co/Z29w1rXdPM
RT @MahilaCongress: Brilliant and brave reporting called #PaisaPolitics by @nit_set on how govt subverted democracy, violated vital anti-mo…
Modi govt violated a vital anti-money laundering rule by allowing anonymous donors to donate expired electoral bond… https://t.co/jCTbANqmn4
RT @RiaRevealed: Finance Ministry told SBI to accept expired electoral bonds worth Rs 10 crore as a one time exception. The bonds were sold…
RT @RuchiraC: Modi govt violated a vital anti-money laundering rule by allowing anonymous donors to donate expired electoral bonds worth Rs…
RT @samar11: Finance Ministry told SBI to accept expired electoral bonds worth Rs 10 crore as a one time exception. The bonds were sold in…
RT @satyarthbhatt1: #WednesdayThoughts #WednesdayThoughts | Captain #AmolYadav, a pilot, innovative who has built a six seater indigenous…

Negative tweets:
In the maharastra state , is there really, sharad pawar has played,a big game with pm narendra modi, instead , outg… https://t.co/bJYGRcYqxu
RT @pbhushan1: After PMO opened an illegal window for Electoral Bonds,the FM violated a vital anti-money laundering rule by allowing anon d…
Govt Made SBI Accept Expired Electoral Bonds Sold In Illegal Window https://t.co/QlWpCNVaAK
RT @saikatd: The govt forced SBI to accept expired electoral bonds. This is a scandal of epic proportions. The government continues to dodg…
Prime Minister Narendra Modi’s ambition to shape the Indo-Pacific great game will fail unless he gets Sri Lankan Pr… https://t.co/0glja4Zfci
RT @aeropradeep: PM Narendra Modi: Central Pay Commission for 10 lakh Bankers & 5 lakh pensioners by abolishing illegal BPS. - Sign the Pet…

CHAPTER 8

CONCLUSION AND FUTURE SCOPE:

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese.

In this project we tried to show the basic way of classifying tweets into positive or negative category using Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together.

Reference:

[1] Efthymios Kouloumpis and Johanna Moore,IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 3, July 2012

[2] S. Batra and D. Rao, ”Entity Based Sentiment Analysis on Twitter”, Stanford University,2010

[3] Saif M. Mohammad and Xiaodan zhu, Sentiment Analysis on of social media texts:,2014

[4] Afroze Ibrahim Baqapuri, Twitter Sentiment Analysis: The Good the Bad and the OMG!, Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media.2011

[5] Links:

- http://www.ijcaonline.org/research/volume125/number3/dandrea-2015-ijca-905866.pdf
- https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis
- textblob.readthedocs.io/en/dev/_modules/textblob/en/sentiments.html