SENTIMENT ANALYSIS OF TAGS (COVID-19 A PANDEMIC OF THE DECADE) FROM TWITTER DATA

Project work submitted to the Pondicherry University in partial fulfillment of the requirements for the award of the Degree of

MASTER OF SCIENCE
IN
STATISTICS
BY

SUBODH BHARTI
Registration No. **14384059**

Under the guidance of

Dr. VISHNU VARDHAN

Assistant Professor

DEPARTMENT OF STATISTICS

RAMANUJAN SCHOOL OF MATHEMATICAL SCIENCES

PONDICHERRY UNIVERSITY

PUDUCHERRY – 605014

MAY 2020

DEPARTMENT OF STATISTICS

RAMANUJAN SCHOOL OF MATHEMATICAL SCIENCES

CERTIFICATE

It is certified that the contents and form of thesis entitled "Sentiment Analysis of tags (covid-19 a Pandemic of the Decade) From Twitter Data" submitted by **SUBODH BHARTI**, Integrated Masters of Science (I.M.Sc.), **Registration no.- 14384059** Batch 2014-2019, of Department Of Statistics have been found satisfactory for the requirement of the degree.

Supervisor:

Dr. Vishnu Vardhan

Assistant Professor

Head

Department of Statistics

Pondicherry University

Submitted for M.Sc. Degree Examination held on……

Examiner

ACKNOWLEDGEMENTS

I am deeply thankful to my Guide, **Dr. Vishnu Vardhan**, for helping me throughout the course in accomplishing my final project. His guidance, support and motivation enabled me in achieving the objectives of the project. And whose role as project guide was invaluable for the project. I am thankful again for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us.

Last but not the least I convey my gratitude to all the teachers for providing me the technical skill that will always remain as my asset and to all non-teaching staff for the gracious hospitality they offered us.

TABLE OF CONTENTS

ABSTRACT

This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users 25M- out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day 25M. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.

PROBLEM DEFINITION:

"Sentiment analysis (also known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information."[1]

Sentiment analysis in the domain of micro-blogging (such as tweets) is a relatively new research topic so still there is a lot of scope for further exploration in this region. A great amount of related previous work is being done in sentiment analysis of web blogs/articles, user's reviews and general phrase level analysis . They are different from twitter primarily because of the limit of 140 characters per tweet which allows the user to express opinion compressed in a very short text. The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on unsupervised and semi-supervised approaches, but still there is need of a lot of improvements. Many researchers has been testing new features and classification techniques just compare their results to base-line performance.

"Sentiment Analysis - 'Is a given piece of text positive, negative, or neutral?'

The text may be a sentence, a tweet, an SMS message, a

customer review, a document, and so on."[2]

CCS Concepts • Information systems→Database management system engines. Keywords Social media; sentiment analysis; twitter data; text mining. 1.

[1] https://en.wikipedia.org/wiki/Sentiment_analysis (Date- 25th June, 2020)

[2] http://www.saifmohammad.com/WebDocs/EMNLP2014-SentimentTutorial.pdf A tutorial presented at the 2014 Conference on Empirical Methods on Natural Language Processing, October 2014, Doha, Qatar.

Chapter 1: INTRODUCTION

Now a days twitter, facebook, whatsapp are getting so much attention from people and also they are getting very much popular among people. Sentiment analysis provides many opportunities to develop a new application. in the industrial field, sentiment analysis has big effect, like government organization and big companies, their desire is to know about what people think about their product, their market value. the aim of sentiment analysis is to find out the mood, behaviour and opinion of person from texts. for the sentiment analysis purpose, social networking used the various sentiment analysis techniques to take the public data. Sentiment analysis widely used in various domain such as finance, economics ,defence , politics. The data available on the social networking sites can be unstructured and structured. almost 80% data on the internet is unstructured. Sentiment analysis techniques are used to find out the people opinion on social media. Twitter is also a huge platform in that different idea, thought, opinion are presented and exchanged. It does not matter where people came from, what religious opinions they hold, rich or poor, educated or uneducated, they comment, compliment, discuss, argue, insist.

1.1 Literature Review:

Sentiment analysis is a growing area of Natural Language Processing (NLP) with research ranging from document level classification (Pang and Lee 2008) to learning the polarity of words and phrases (e.g., (Hatzivassiloglou and McKeown 1997; Esuli and Sebastiani 2006)). Given the character limitations on tweets, classifying the sentiment of Twitter messages is most similar to sentencelevel sentiment analysis (e.g., (Yu and Hatzivassiloglou 2003; Kim and Hovy 2004)); however, the informal and specialized language used in tweets, as well as the very nature of the microblogging domain make Twitter sentiment analysis a very different task. It's an open question how well the features and techniques used on more well-formed data will transfer to the microblogging domain.

Earlier there have been a number of papers looking at Twitter sentiment and buzz (Jansen et al. 2009 ; Pak and Paroubek 2010; O'Connor et al. 2010; Tumasjan et al. 2010; Bifet and Frank 2010; Barbosa and Feng 2010 ; Davidov, Tsur, and Rappoport 2010). Further researchers have started to explore the use of 'parts-of-speech' structures but results stay mixed. Structures common to microblogging (e.g., emoticons) are also frequent, on the other way there has been little study done in the usefulness of non-microblogging data.

Researchers also started to explore various ways of automatically collecting 'training data'. Several researchers dependent on 'emoticons' for defining their training data ("Pak and Paroubek 2010; Bifet and Frank 2010"). ('Barbosa and Feng 2010') exploit existing Twitter sentiment sites for collecting training data. ("Davidov, Tsur, and Rappoport 2010") also usage hashtags for generating training data, but they bound their experiments to sentiment/non-sentiment classification, rather than 3-way polarity classification, as we do. We use WEKA and implement the following Machine Learning algorithms for second classification to arrive at the best result:

- K-Means Clustering
- K Nearest Neighbours
- Support Vector Machine
- Logistic Regression
- Rule Based Classifiers
- Naive Bayes

1.2 FUNCTIONALITY AND DESIGN

We used Naïve Bayes Classifiers to fetch Tweets from Twitter Data.

• **Naive Bayes Classification:**

"Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x1 through xn, :

$P(y|x1,\ldots,xn)=P(y)P(x1,\ldots,xn|y)P(x1,\ldots,xn)$

Using the naive conditional independence assumption that

$P(xi|y,x1,\ldots,xi-1,xi+1,\ldots,xn)=P(xi|y)$,

for all i, this relationship is simplified to

$P(y|x1,\ldots,xn)=P(y)\prod i=1nP(xi|y)P(x1,\ldots,xn)$

Since P(x1,…,xn) is constant given the input, we can use the following classification rule:

$P(y|x1,\ldots,xn)\propto P(y)\prod i=1nP(xi|y)\Downarrow y\hat{}=arg\,maxyP(y)\prod i=1nP(xi|y)$,

and we can use Maximum A Posteriori (MAP) estimation to estimate P(y) and P(xi|y); the former is then the relative frequency of class y in the training set.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi|y).

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why naive Bayes works well, and on which types of data it does, see the references below.)

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality."[3]

[3] https://scikit-learn.org/stable/modules/naive_bayes.html (Date- June 25, 2020)

For example 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:

+ ...z any 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…

SRS (Software Requirement Specification):

• Internal Interface requirement: Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem. Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority. Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type. Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. The recent explosion in data pertaining to users on social media has created a great interest in performing sentiment analysis on this data using Big Data and Machine Learning principles to understand people's interests. This project intends to perform the same tasks. The difference between this project and other sentimnt analysis tools is that, it will perform real time analysis of tweets based on hashtags and not on a stored archive.

;Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a ;follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the ;SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this ;software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, ;subsystem interconnections, and external interfaces can be helpful.

The Product functions are: • Collect tweets in a real time fashion i.e. , from the twitter live stream based on specified hashtags

• Remove redundant information from these collected tweets.

• Store the formatted tweets in MongoDB database

• Perform Sentiment Analysis on the tweets stored in the database to classify their nature viz. positive, negative and so on.

• Use a machine learning algorithm which will predict the 'mood' of the people with respect ot that topic.

;Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, ;so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to ;any reader of the SRS. A picture of the

major groups of related requirements and how they relate, such as a top level data ;flow diagram or object class diagram, is often effective.

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy. Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

• External Interface Requirement:

We classify External Interface in 4 types, those are: User Interface: Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Hardware interface:

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

Software Interface:

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components.

Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for

example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

- Os- Linux Operating System/Windows 10 (2019 version)/Mac Os
- Python- Python via Anaconda version 3.7
- Browser based IDE for Python : Jupyter Notebook server Version 6.0.0, IDE spyder 3.7
- Modern Web Browser like- Internet Explorer, Google Chrome, Firefox Mozilla etc.
- (open source free of charge url- https://www.anaconda.com/products/individual#windows)
- Twitter User account
- Twitter App Developer User id account

List of Libraries for Sentiment analysis of Twitter Data for Python

- numpy
- pandas
- re (Regular Expression)
- string
- nltk
- Sklearn
- Keras
- csv
- tweepy
- textblob
- matplotlib
- random

Communication Interface:

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

1.3 NON FUNCTIONAL REQUIREMENT:

Performance Requirements:

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

Safety Requirements:

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

Security Requirements:

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

Software Quality Attributes:

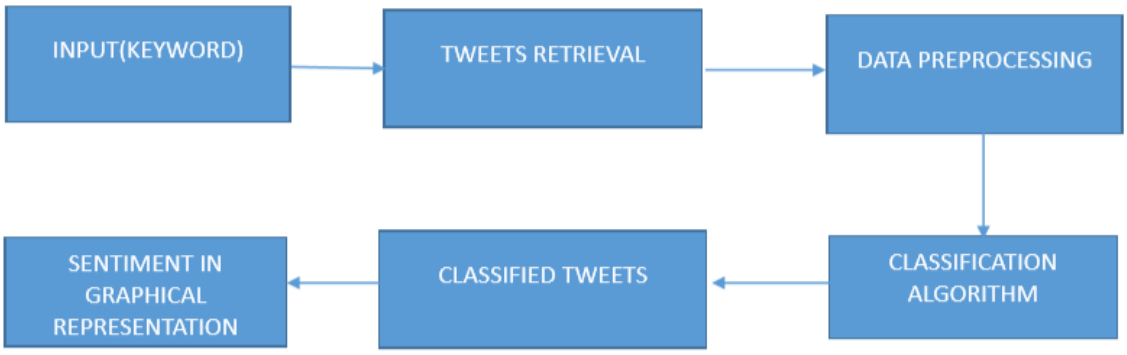
Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At

the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

Other Requirements:

- Linux Operating System/Windows
- Python Platform(Anaconda2,Spyder,Jupyter)
- NLTK package,
- Modern Web Browser
- Twitter API, Google API

Design:


INPUT(KEYWORD)
TWEETS RETRIEVAL
DATA PREPROCESSING
CLASSIFICATION ALGORITHM
CLASSIFIED TWEETS
SENTIMENT IN GRAPHICAL REPRESENTATION

Chapter 2: Data and Methodology

2.1 KeyWord/ Tags/ Hashtags/ search_query:

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".

2.2 TWEETS RETRIEVAL 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.

2.3 CLASSIFICATION ALGORITHM:

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. var algorithmia = require("algorithmia");

```
var client = algorithmia(process.env.ALGORITHMIA_API_KEY);

var input = "happy"; var no_retweets = [];


console.log("Analyzing tweets with phrase: " + input);

 client.algo("/diego/RetrieveTweetsWithKeyword/0.1.2").pipe(input).then(function(output)

{    if (output.error) {       console.log(output.error);

 }

else {       var tweets = [];       var tweets = output.result;

 for (var i = 0; i < output.result.length; i++)

{         // Remove retweets. All retweets contain "RT" in the string.

 if (tweets[i].indexOf('RT') == -1)

{           no_retweets.push(tweets[i]);

  }    } }


// We will cover this function in the next step. analyze_tweets(no_retweets); });
```

2.4 Code to Fetch the Twitter Data based on a query/search_words/hashtags/tags/keywords

```
import tweepy

# authorization tokens
consumer_key = 'jbkZ2RmduoH7eh9WBpPUQWro3'
consumer_secret = 'RpdmaKXShnzz8NdjlBG6vad88E3CebNtx5Gb03fkEvGCst1fIo'
access_key = '585362387-kqPJV0ztt0Q1RVQdz77LTMhWV0CzNgk1cAJwmBHm'
access_secret = 'jYfkksmxZybdn9pwpArAVWzSDpe0J4yaafU3EqQS9yJQ3'
'''consumer_key = "[insert your key here]"
consumer_secret = "[insert your secret here]"
access_key = "[insert your key here]"
access_secret = "[insert your secret here]"
'''
# StreamListener class inherits from tweepy.StreamListener and overrides on_status/on_error methods.
class StreamListener(tweepy.StreamListener):
    def on_status(self, status):
        print(status.id_str)
        # if "retweeted_status" attribute exists, flag this tweet as a retweet.
        is_retweet = hasattr(status, "retweeted_status")

        # check if text has been truncated
        if hasattr(status,"extended_tweet"):
            text = status.extended_tweet["full_text"]
        else:
            text = status.text

        # check if this is a quote tweet.
        is_quote = hasattr(status, "quoted_status")
```

```
        quoted_text = ""
        if is_quote:
            # check if quoted tweet's text has been truncated before recording it
            if hasattr(status.quoted_status,"extended_tweet"):
                quoted_text = status.quoted_status.extended_tweet["full_text"]
            else:
                quoted_text = status.quoted_status.text

        # remove characters that might cause problems with csv encoding
        remove_characters = [",","\n"]
        for c in remove_characters:
            text.replace(c," ")
            quoted_text.replace(c, " ")

        with open("out1.csv", "a", encoding='utf-8') as f:
            f.write("%s,%s,%s,%s,%s,%s\n"                                                    %
(status.created_at,status.user.screen_name,is_retweet,is_quote,text,quoted_text))

    def on_error(self, status_code):
        print("Encountered streaming error (", status_code, ")")
        sys.exit()

if __name__ == "__main__":
    # complete authorization and initialize API endpoint
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)

    # initialize stream
```

```
    streamListener = StreamListener()
    stream = tweepy.Stream(auth=api.auth, listener=streamListener,tweet_mode='extended')
    with open("out.csv", "w", encoding='utf-8') as f:
        f.write("date,user,is_retweet,is_quote,text,quoted_text\n")
    tags = ["covid19"]
    stream.filter(track=tags,)

# Collect tweets
tweets = tw.Cursor(api.search,
            q=search_words,
            lang="en",
            since=date_since).items(5)

# Iterate and print tweets
for tweet in tweets:
    print(tweet.text)
new_search = "climate+change -filter:retweets"

tweets = tw.Cursor(api.search,
                q=new_search,
                lang="en",
                since='2018-04-23').items(1000)

all_tweets = [tweet.text for tweet in tweets]
all_tweets[:5]
```

In the above algorithem, we iterated through each tweet in no_retweets to send that as input to the Sentiment Analysis algoritm. Then with the results from that API call, we added the output result to a total_score variable. We keep track of how many tweets we've gone through with the variable score_count, so that when it reaches the same number as the number of tweets we wanted to analyze we then calculate the final score by averaging the total_score. This final result returns a number in the range [0-4] representing, in order, very negative, negative, neutral, positive, and very positive sentiment.

2.5 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".

<Blank>: Leave the tweet unlabelled if it belongs to some language other than English so that it is ignored in the training data. Now that we have discussed some of the text formatting techniques employed by us, we will move to the list of features that we have explored. As we will see below a feature is any variable which can help our classifier in differentiating between the different classes. There are two kinds of classification in our system (as will be discussed in detail in the next section), the objectivity / subjectivity classification and the positivity / negativity classification. As the name suggests the former is for differentiating between objective and subjective classes while the latter is for differentiating between positive and negative classes. The list of features explored for objective / subjective classification is as below:

- Number of exclamation marks in a tweet
- Number of question marks in a tweet
- Presence of exclamation marks in a tweet
- Presence of question marks in a tweet
- Presence of url in a tweet
- Presence of emoticons in a tweet
- Unigram word models calculated using Naive Bayes
- Prior polarity of words through online lexicon MPQA

- Number of digits in a tweet
- Number of capitalized words in a tweet
- Number of capitalized characters in a tweet
- Number of punctuation marks / symbols in a tweet
- Ratio of non-dictionary words to the total number of words in the tweet
- Length of the tweet Number of adjectives in a tweet
- Number of comparative adjectives in a tweet
- Number of superlative adjectives in a tweet
- Number of base-form verbs in a tweet
- Number of past tense verbs in a tweet
- Number of present participle verbs in a tweet
- Number of past participle verbs in a tweet
- Number of 3rd person singular present verbs in a tweet
- Number of non-3rd person singular present verbs in a tweet
- Number of adverbs in a tweet
- Number of personal pronouns in a tweet
- Number of possessive pronouns in a tweet
- Number of singular proper noun in a tweet
- Number of plural proper noun in a tweet
- Number of cardinal numbers in a tweet
- Number of possessive endings in a tweet
- Number of wh-pronouns in a tweet
- Number of adjectives of all forms in a tweet
- Number of verbs of all forms in a tweet
- Number of nouns of all forms in a tweet
- Number of pronouns of all forms in a tweet

The list of features explored for positive / negative classification are given below

- Overall emoticon score (where 1 is added to the score in case of positive emoticon, and 1 is subtracted in case of negative emoticon)
- Overall score from online polarity lexicon MPQA (where presence of strong positive word in the tweet increases the score by 1.0 and the presence of weak negative word would decrease the score by 0.5)
- Unigram word models calculated using Naive Bayes
- Number of total emoticons in the tweet
- Number of positive emoticons in a tweet
- Number of negative emoticons in a tweet
- Number of positive words from MPQA lexicon in tweet
- Number of negative words from MPQA lexicon in tweet
- Number of base-form verbs in a tweet
- Number of past tense verbs in a tweet
- Number of present participle verbs in a tweet

- Number of past participle verbs in a tweet
- Number of 3rd person singular present verbs in a tweet
- Number of non-3rd person singular present verbs in a tweet
- Number of plural nouns in a tweet
- Number of singular proper nouns in a tweet
- Number of cardinal numbers in a tweet
- Number of prepositions or coordinating conjunctions in a tweet
- Number of adverbs in a tweet
- Number of wh-adverbs in a tweet
- Number of verbs of all forms in a tweet

2.6 Python Code for sentiment Analysis on Twitter Data of COVID-19 Pandemic 2020

```
“import re
import csv
import numpy as np
import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob
import matplotlib.pyplot as plt
import random
import string
from nltk.corpus import wordnet as wn
#import preprocessor as p
#p.clean('Preprocessor is #awesome 👍 https://github.com/s/preprocessor')

class TwitterClient(object):
    def __init__(self):
        consumer_key = 'jbkZ2RmduoH7eh9WBpPUQWro3'
        consumer_secret = 'RpdmaKXShnzz8NdjlBG6vad88E3CebNtx5Gb03fkEvGCst1flo'
        access_token = '585362387-kqPJV0ztt0Q1RVQdz77LTMhWV0CzNgk1cAJwmBHm'
        access_token_secret = 'jYfkksmxZybdn9pwpArAVWzSDpe0J4yaafU3EqQS9yJQ3'
        try:
            self.auth = OAuthHandler(consumer_key, consumer_secret)
            self.auth.set_access_token(access_token, access_token_secret)
            self.api = tweepy.API(self.auth)
        except:
            print("Error: Authentication Failed")
```

```
    def clean_tweet(self, tweet):
        sent_tweet=' '.join(re.sub(
            "(@[A-Za-z0-9]+)|([^0-9A-Za-z
\t])|(\w+:\/\/\S+)|positive|Positive|POSITIVE|PoStive|negative|Negative|NEGATIVE",
                                " ", tweet).split())
        return sent_tweet


    def get_tweet_sentiment(self, tweet):
        analysis = TextBlob(self.clean_tweet(tweet))
        if analysis.sentiment.polarity > 0:
            return 'positive'
        elif analysis.sentiment.polarity == 0:
            return 'neutral'
        else:
            return 'negative'
    def get_tweets(self, query, count = 20000):
        tweets = []
        try:
            fetched_tweets = self.api.search(q = query + '-filter:retweets -filter:replies' , count = count)
            for tweet in fetched_tweets:
                parsed_tweet = {}
                parsed_tweet['text'] = tweet.text
                parsed_tweet['sentiment'] = self.get_tweet_sentiment(tweet.text)
                if tweet.retweet_count > 0:
                    if parsed_tweet not in tweets:
                        tweets.append(parsed_tweet)
                else:
                    tweets.append(parsed_tweet)
            return tweets
```

```
        except tweepy.TweepError as e:
            print("Error : " + str(e))
def plot():
    api = TwitterClient()
    print('enter queries \n')
    q1=input('enter queries with OR logical for multiple else simple space for AND logic \n ')
    q=''
    for ss in wn.synsets(q1): # Each synset represents a diff concept.(synonyms of the word)
        q=ss.lemma_names()
        #print( ss.lemma_names())

    # Python program to convert a list to string using list comprehension. replace space with OR logic
    listToStr = ' '.join([str(elem) for elem in q])
    q=listToStr
    q=q.replace(' ', ' OR ')
    if q!='':
        q=q1+" OR "+q
    q=q.replace(' OR OR ',' OR ' )
    #print(q)
    n=input('enter number of tweets to analyze \n')



    tweets = api.get_tweets(query = q + '-filter:retweets -filter:replies',count = n)
    ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']

    pt=format(100*len(ptweets)/len(tweets))
    print("POSITIVE TWEET PERSENTAGE: ",pt)
```

```
ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']

#for tweets of neutral sentiment
nuttweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']

nt=format(100*len(ntweets)/len(tweets))
print("NEGATIVE TWEET PERSENTAGE: ",nt)

nut=format(100*(len(tweets) - len(ntweets) - len(ptweets))/len(tweets))
print("NEUTRAL TWEET PERSENTAGE",nut)

#for pie chart
l = ['Positive', 'Negative', 'Neutral']
sizes = [pt,nt,nut]
colors = ['green', 'red', 'blue']
explode = (0.1,0,0) # explode 1st slice
#textprops = {"fontsize":50} # Font size of text in pie chart
#plt.setp(autotexts, size=8, weight="bold")

#patches, texts =
fig = plt.figure(figsize =(12, 7))

#plt.pie(data, labels = cars)
plt.pie(sizes, explode=explode, labels=l, colors=colors, textprops={'color':"lawngreen", 'fontsize':13},
      autopct='%1.1f%%', shadow=True, startangle=45)
'''plt.pie(sizes, explode=explode, labels=labels, colors=colors,
         autopct='%1.1f%%', shadow=True, startangle=40,textprops =textprops)
'''
```

```
    plt.axis('equal')

    # random string without repeating letters
    def randomString2(stringLength):
        letters = string.ascii_lowercase
        return ''.join(random.sample(letters, stringLength))
    myfig=randomString2(6)
    plt.savefig( 'myfig' )
    #print('myfig.png')
    plt.title('Sentiment Analysis Pie chart')
    plt.show()




    print("\n\nPOSITIVE TWEETS:")
    for tweet in ptweets[:10]:
        print(tweet['text'])
    print("\n\nNEGATIVE TWEETS:")
    for tweet in ntweets[:10]:
        print(tweet['text'])
    print("\n\nNeutral TWEETS:")
    for tweet in nuttweets[:10]:
        print(tweet['text'])


def main():
    plot()
if __name__ == "__main__":
    main() ”
```

Chapter 3: Analysis and Results

We will first present our results for the objective / subjective and positive / negative classifications. These results act as the first step of our classification approach. We only use the short-listed features for both of these results. This means that for the objective / subjective classification we have 5 features and for positive / negative classification we have 3 features. For both of these results we use the Naïve Bayes classification algorithm, because that is the algorithm we are employing in our actual classification approach at the first step. In final classification approach, condition is removed and basically both objectivity and polarity classifications are applied to all tweets regardless of whether they are labelled objective or subjective.

(i) For number of tweets 100 to analyse, we got Results as :positive tweets percentage 9.0%, Negative tweets percentage 5.0% and Neutral tweets percentage 86.0%

(ii) For number of tweets 1000 to analyse, we got Results as :positive tweets percentage 13.0%, Negative tweets percentage 8.0% and Neutral tweets percentage 79.0%

(iii) For number of tweets 10k, to analyse , we got Results as :positive tweets percentage 7.0%, Negative tweets percentage 2.0% and Neutral tweets percentage 91.0%

(iv) For number of tweets 1 million to analyse, we got Results as :positive tweets percentage 10.0%, Negative tweets percentage 2.0% and Neutral tweets percentage 88.0%

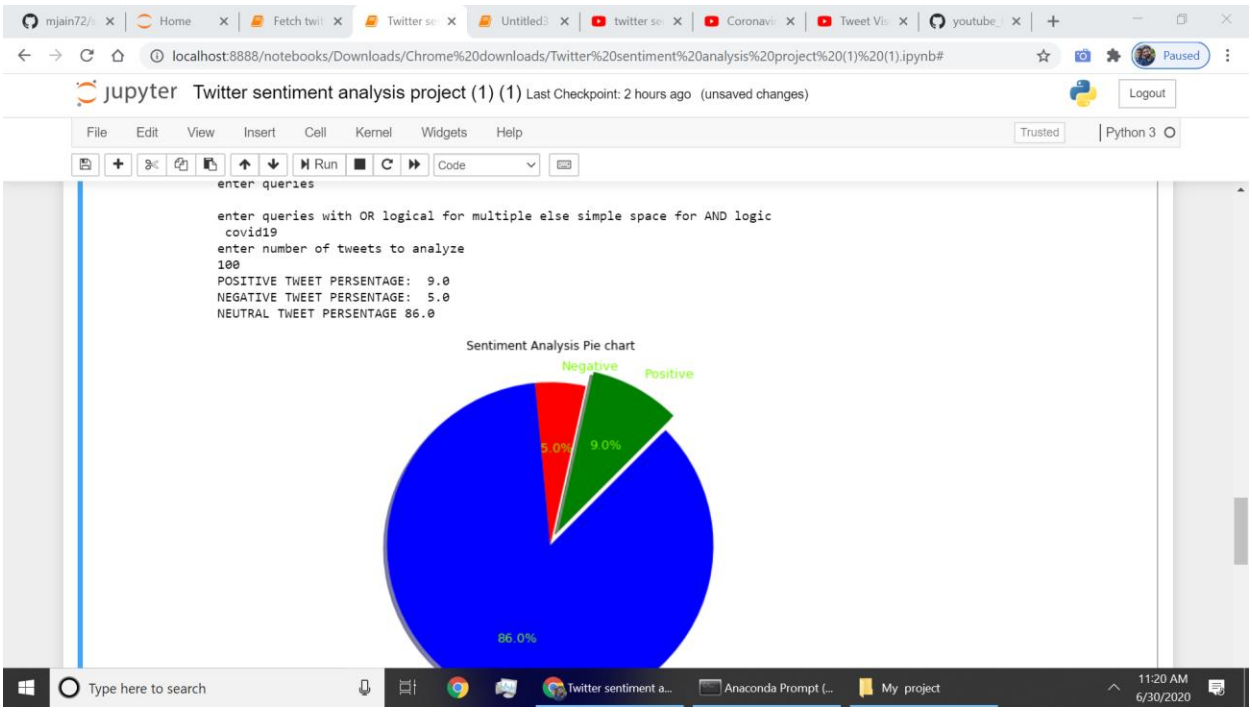
(v) For number of tweets 10 million to analyse, we got Results as :positive tweets percentage 14.0%, Negative tweets percentage 6.0% and Neutral tweets percentage 80.0%

On the basis of above sentiment through python libraries using Naïve base algorithm we can say people are slightly positive toward covid -19 pandemic of the Decade, ignoring the fact that percentage of tweets lies far higher in Neutral sentiments. It might have more possibility that some less or from the tweets percentage from Neutral Sentiment might support Positive sentiment or might support negative sentiment towards Covid-19 Pandemic of the Decade.

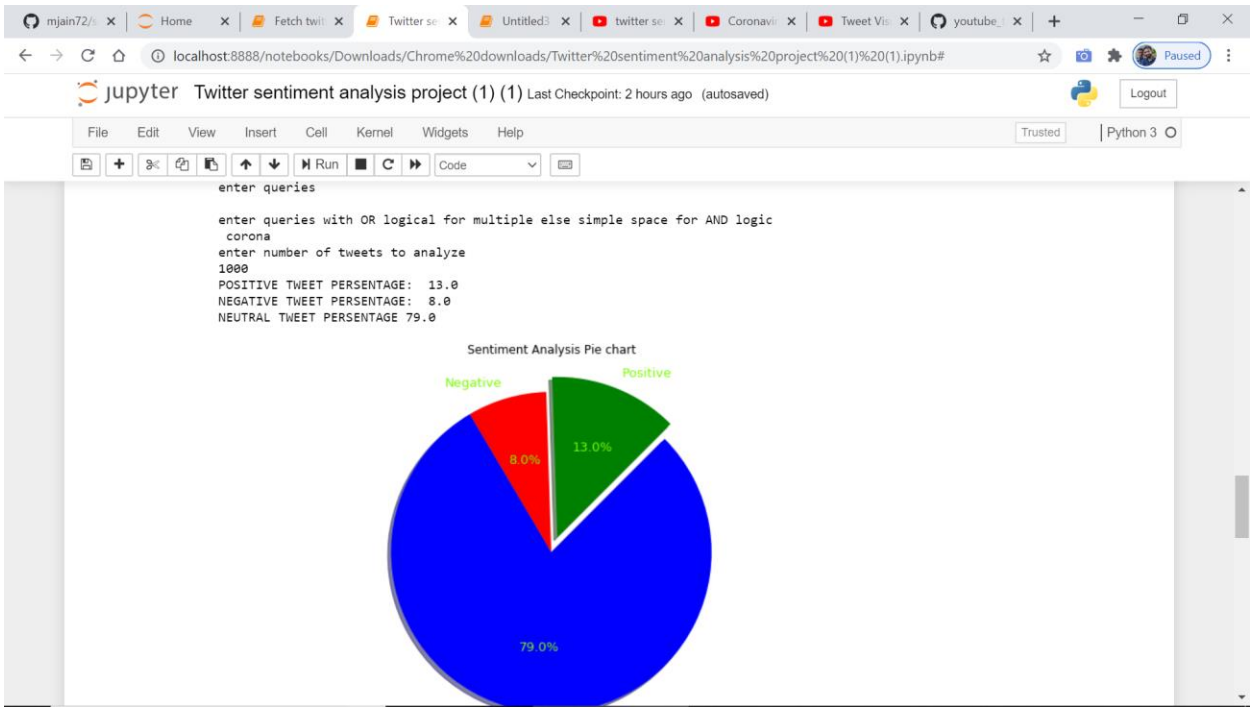
Still There is need of a lot improvements required and there is scope of new algorithms with improved accuracy might help a lot in future.

3.1 Figures reported are the result:

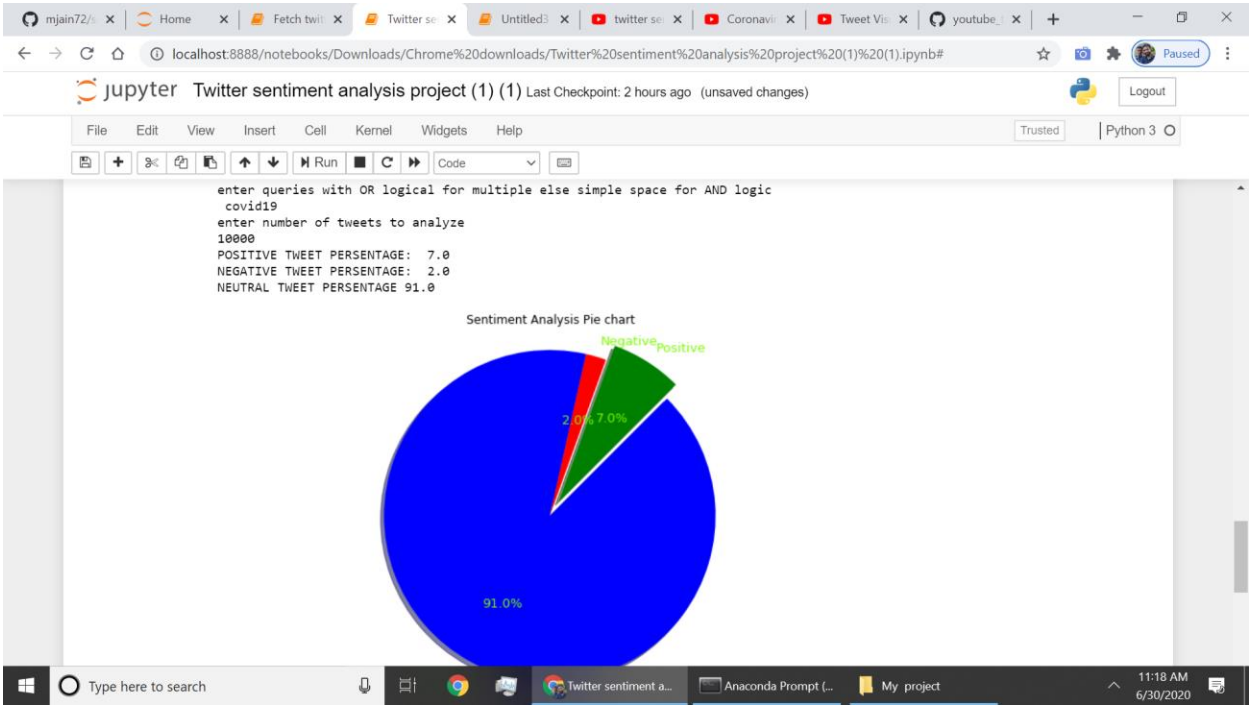
Furthermore all the figures reported are the result:



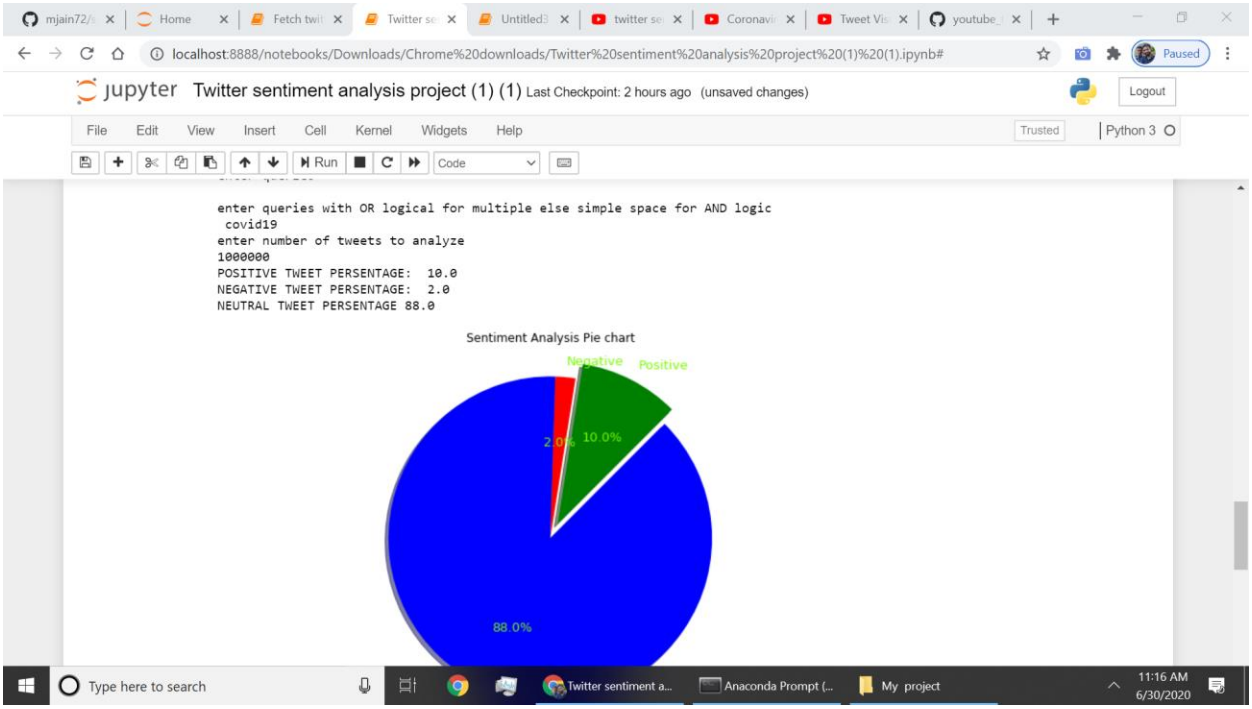
For number of tweets 100 to analyse, we got Results as :positive tweets percentage 9.0%, Negative tweets percentage 5.0% and Neutral tweets percentage 86.0%



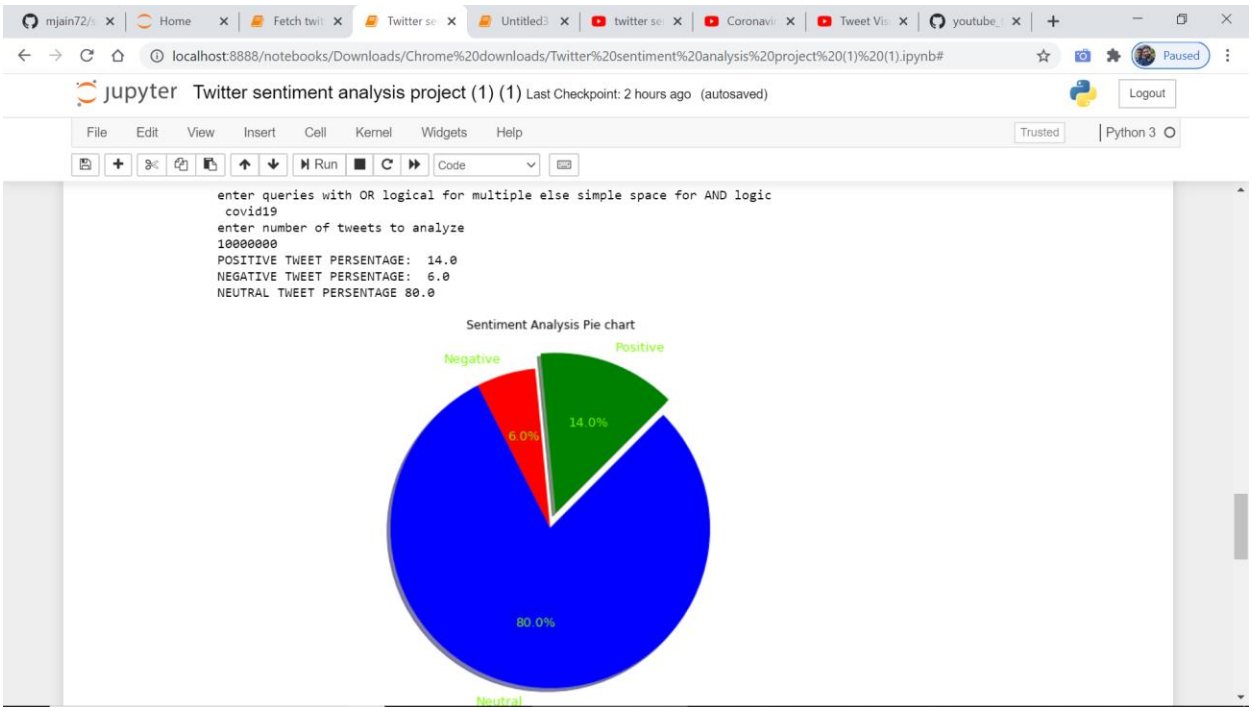
For number of tweets 1000 to analyse, we got Results as :positive tweets percentage 13.0%, Negative tweets percentage 8.0% and Neutral tweets percentage 79.0%



For number of tweets 10k, to analyse , we got Results as :positive tweets percentage 7.0%, Negative tweets percentage 2.0% and Neutral tweets percentage 91.0%



For number of tweets 1 million to analyse, we got Results as :positive tweets percentage 10.0%, Negative tweets percentage 2.0% and Neutral tweets percentage 88.0%



For number of tweets 10 million to analyse, we got Results as :positive tweets percentage 14.0%, Negative tweets percentage 6.0% and Neutral tweets percentage 80.0%

Chapter 4: Conclusion and Future Scope:

The task of sentiment analysis, especially in micro-bloging, is still in the developing stage and far from complete. Still There is need of a lot improvements required and there is scope of new algorithms with improved accuracy might help a lot in future. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance.

Now we have worked with only the very simplest unigram models; we can improve those models by adding extra information like closeness of the word with a negation word. We could specify a window prior to the word (a window could for example be of 2 or 3 words) considering the effect of negation may be incorporated into the model if it lies within that window. The closer the negation word is to the unigram word whose prior polarity is to be calculated, the more it would be affected the polarity. For example if the negation is right next to the word, it may simply reverse the polarity of that word and farther the negation is from the word the more minimized ifs effect should be.

Other feature that is worth exploring is whether the information about relative position of word in a tweet has any effect on the performance of the classifier. Though It is explored a similar feature and reported negative results, their results were based on reviews which are very different from tweets and they worked on an very simple model.

In this project we are focussing on general sentiment analysis. There is potential of work in the field of sentiment analysis with partially known context. We noticed that users generally use websites for specific types of keywords which can divided into a couple of distinct classes, namely: politics/politicians, celebrities, products/brands, sports/sportsmen, media/movies/music. That might be an attempt to perform separate sentiment analysis on tweets that only belong to one of these classes (i.e. the training data would not be general but specific to one of these categories) and compare the results we get i f we apply general sentiment analysis on it instead.

REFERENCES:

Efthymios Kouloumpis and Johanna Moore, IJCSI International Journal of Computer Science Issues, Vol. 9,

Issue 4, No 3, July 2012

http://www.saifmohammad.com/WebDocs/EMNLP2014-SentimentTutorial.pdf A tutorial presented at the 2014 Conference on Empirical Methods on Natural Language Processing, October 2014, Doha, Qatar.

S. Batra and D. Rao, "Entity Based Sentiment Analysis on Twitter", Stanford University, 2010

Ekaterina kochmar, University of Cambridge, at the Cambridge coding Academy Data Science. 2016

Manju Venugopalan and Deepa Gupta , Exploring Sentiment Analysis on Twitter Data, IEEE 2015

Brett Duncan and Yanqing Zhang, Neural Networks for Sentiment Analysis on Twitter. 2017

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

Some other sources :

Online Study materials, content and help- Youtube, GitHub, Dataquest.io, Python official site for Documentations. And many more .