## Program Flow:

The project involves Connecting to Twitter API, Collecting tweets using the keys obtained, Pre- processing of the data, Classifying of tweets using Naïve Bayes Classifier, Analysing the final output by drawing a pie chart which shows the percentages of categories.

## Sample Code:

### Code 1: Streaming of twitter data on the hash tags ‘iPhone News’, ‘apple’

from tweepy.streaming import StreamListener from tweepy import OAuthHandler

from tweepy import Stream import time

#Variables that contains the user credentials to access Twitter API

access\_token = "4660193893-GEzV6ztLgWDzkTcV3qDed3VZp6GRwe1bEw47YPm" access\_token\_secret = "zvi13Gbi64Xj4JcPE3sDROJRFnbGf5IKpDo550ZgrUdNn" consumer\_key = "M1HmAEnxLMp45v6TT7QljUqqs"

consumer\_secret = "bkELGnxVoHngs8GSwbQ9xdiHkFiB4J9JMcWPyRJS5360jDPtuS" #This is a basic listener that just prints received tweets to stdout.

class StdOutListener(StreamListener):

def on\_data(self, data): try:

print data

saveFile = open('iPhn.txt','a') saveFile.write(data)

saveFile.write('\n') saveFile.close()

return True

except BaseException, e:

print 'failed ondata,', str(e) time.sleep(5)

def on\_error(self, status): print status

if name == ' main ':

#This handles Twitter authentification and the connection to Twitter Streaming API l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret) auth.set\_access\_token(access\_token, access\_token\_secret) stream = Stream(auth, l)

#This line filter Twitter Streams to capture data by the keywords: 'python', 'javascript', 'ruby' stream.filter(track=['iPhone\_News', 'appleios', 'iPhone News'])

**Code 2: Converting the obtained streamed data into JSON format for further process. JSON (JavaScript Object Notation) is a lightweight format that is used for data interchanging.**

try:

import json except ImportError:

import simplejson as json

# We use the file saved from last step as example tweets\_filename = 'iPhn.txt'

tweets\_file = open(tweets\_filename, "r") for line in tweets\_file:

try:

# Read in one line of the file, convert it into a json object tweet = json.loads(line.strip())

if 'text' in tweet: # only messages contains 'text' field is a tweet print tweet['id'] # This is the tweet's id

print tweet['created\_at'] # when the tweet posted print tweet['text'] # content of the tweet

print tweet['user']['id'] # id of the user who posted the tweet print tweet['user']['name'] # name of the user, e.g. "Wei Xu"

print tweet['user']['screen\_name'] # name of the user account, e.g. "cocoweixu" print

hashtags = []

for hashtag in tweet['entities']['hashtags']: hashtags.append(hashtag['text'])

print hashtags

except:

# read in a line is not in JSON format (sometimes error occured) continue

### Code 3: Removing the URLs’ and other IDs’ related to the tweets

from nltk.tokenize import word\_tokenize from tweepy.streaming import StreamListener from tweepy import OAuthHandler

from tweepy import Stream import json

import re import string

file = open('json2.txt', 'r') for line in file:

#matching of urls’ using RE library

tweet = re.match('(.\*?)http.\*?\s?(.\*?)', line) #token\_tweet=word\_tokenize(tweet)

if tweet:

print tweet.group(1) print tweet.group(2)

#if token\_tweet:

#print token\_tweet.group(1)

### Code 4: Removing punctuations and other symbols which are unnececsary for the data classification

from nltk.tokenize import word\_tokenize from nltk.corpus import stopwords

from tweepy.streaming import StreamListener from tweepy import OAuthHandler

from tweepy import Stream import json

import re import string

file = open('jsonF.txt', 'r') for word in file:

for punct in string.punctuation: word = word.replace(punct,"")

print word

### Code 5: Removing stopwords (eg: a, about, is, to etc..)

from nltk.tokenize import word\_tokenize from nltk.corpus import stopwords

from tweepy.streaming import StreamListener from tweepy import OAuthHandler

from tweepy import Stream import json

import re

import string

file = open('rem\_puncF.txt', 'r') filtered=[]

for line in file:

stop\_words=set(stopwords.words("english")) if line not in stop\_words:

filtered.append(line) print line

words=word\_tokenize(line)

filtered\_line = [w for w in line.split() if not w in stop\_words] filtered\_line=' '.join(filtered\_line)+'\n'

print filtered\_line #for w in words:

# if w not in stop\_words:

# filtered.append(w)

# print (''.join(w)) #print('\n')

### Code 6: Removing excess spaces and obtaining cleaned tweets finally

from tweepy.streaming import StreamListener from tweepy import OAuthHandler

from tweepy import Stream import json

import re

import string f=open('rem\_stopFi.txt', 'r') for line in f:

cleanedLine = line.strip()

if cleanedLine: # is not empty print

print(cleanedLine)

### Code 7: Analysing tweets using Naïve Bayes Classifier and plotting a pie graph using Matplot Library

import nltk

import matplotlib.pyplot as plt

from nltk.tokenize import word\_tokenize short\_pos = open("pos.txt","r").read()

short\_neg = open("neg.txt","r").read() short\_neut = open("neut.txt","r").read() all\_words = []

doc\_neg = [] doc\_pos = [] doc\_neut = []

for p in short\_pos.split('\n'): doc\_pos.append( (p, "pos") )

for p in short\_neg.split('\n'):

doc\_neg.append( (p, "neg") ) for p in short\_neut.split('\n'):

doc\_neut.append( (p, "neut") ) count\_pos=0

count\_neg=0 count\_neut=0 tweets = []

for (words, sentiment) in pos\_tweets + neg\_tweets+neut\_tweets: words\_filtered = [e.lower() for e in words.split() if len(e) >= 3] tweets.append((words\_filtered, sentiment))

def get\_words\_in\_tweets(tweets): all\_words = []

for (words, sentiment) in tweets: all\_words.extend(words)

return all\_words

def get\_word\_features(wordlist): wordlist = nltk.FreqDist(wordlist) word\_features = wordlist.keys() return word\_features

word\_features = get\_word\_features(get\_words\_in\_tweets(tweets)) def extract\_features(document):

document\_words = set(document) features = {}

for word in word\_features:

features['contains(%s)' % word] = (word in document\_words) return features

training\_set = nltk.classify.apply\_features(extract\_features, tweets) classifier = nltk.NaiveBayesClassifier.train(training\_set) f=open('rem\_spaceFi1.txt', 'r')

for line in f: tweet = line

test=classifier.classify(extract\_features(tweet.split())) if test=='positive':

count\_pos=count\_pos+1 elif test=='negative': count\_neg=count\_neg+1 elif test=='neutral': count\_neut=count\_neut+1

print 'count\_pos' print pos

print 'count\_neg' print neg

print 'count\_neut' print neut

total=count\_pos+count\_neg+count\_neut print total

percentage\_pos=pos\*100/total percentage\_neg=neg\*100/total percentage\_neut=neut\*100/total

print 'percentage\_pos' print percentage\_pos

print 'percentage\_neg' print percentage\_neg

print 'percentage\_neut' print percentage\_neut

import matplotlib.pyplot as plt # Data to plot

labels = 'Negative %','positive %','neutral %'

sizes = [percentage\_neg, percentage\_pos, percentage\_neut] colors = ['red', 'green','lightskyblue']

explode = (0.1, 0, 0) # explode 1st slice # Plot

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal') plt.show()

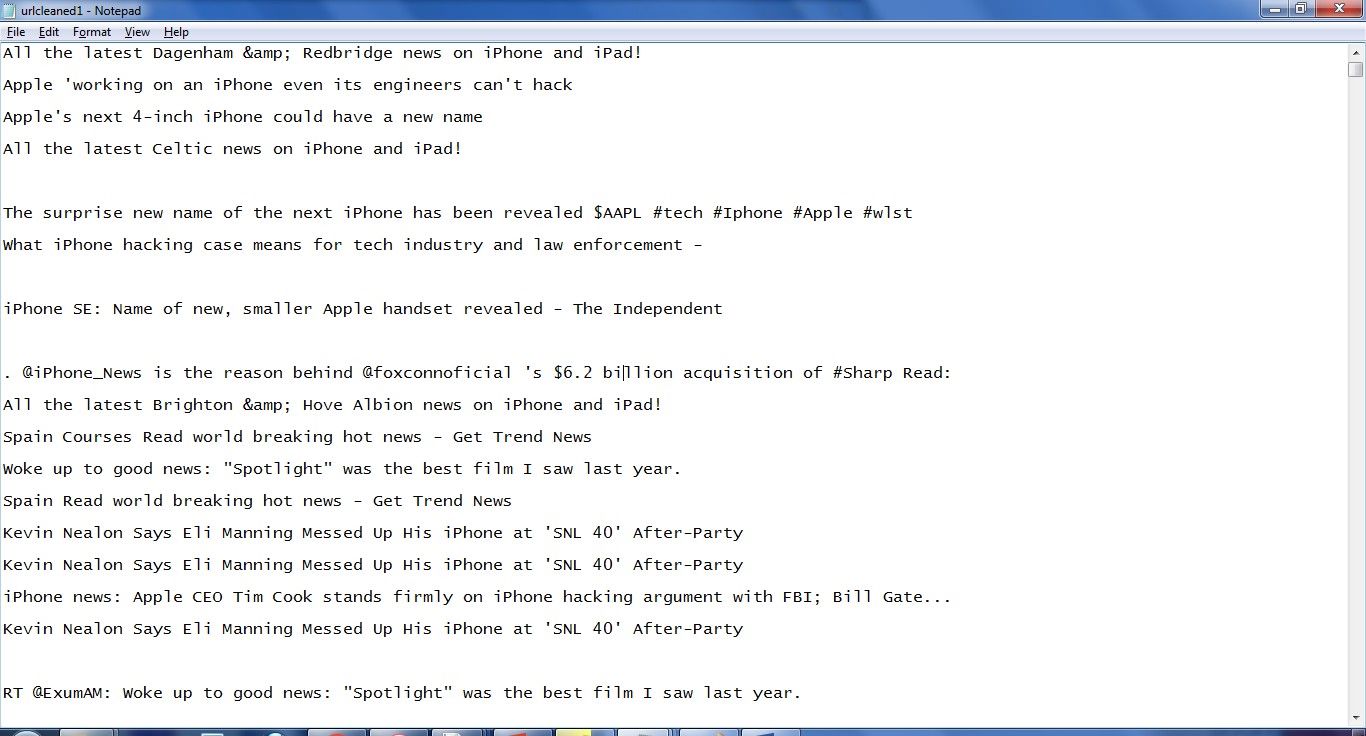
### SCREEN SHOTS:



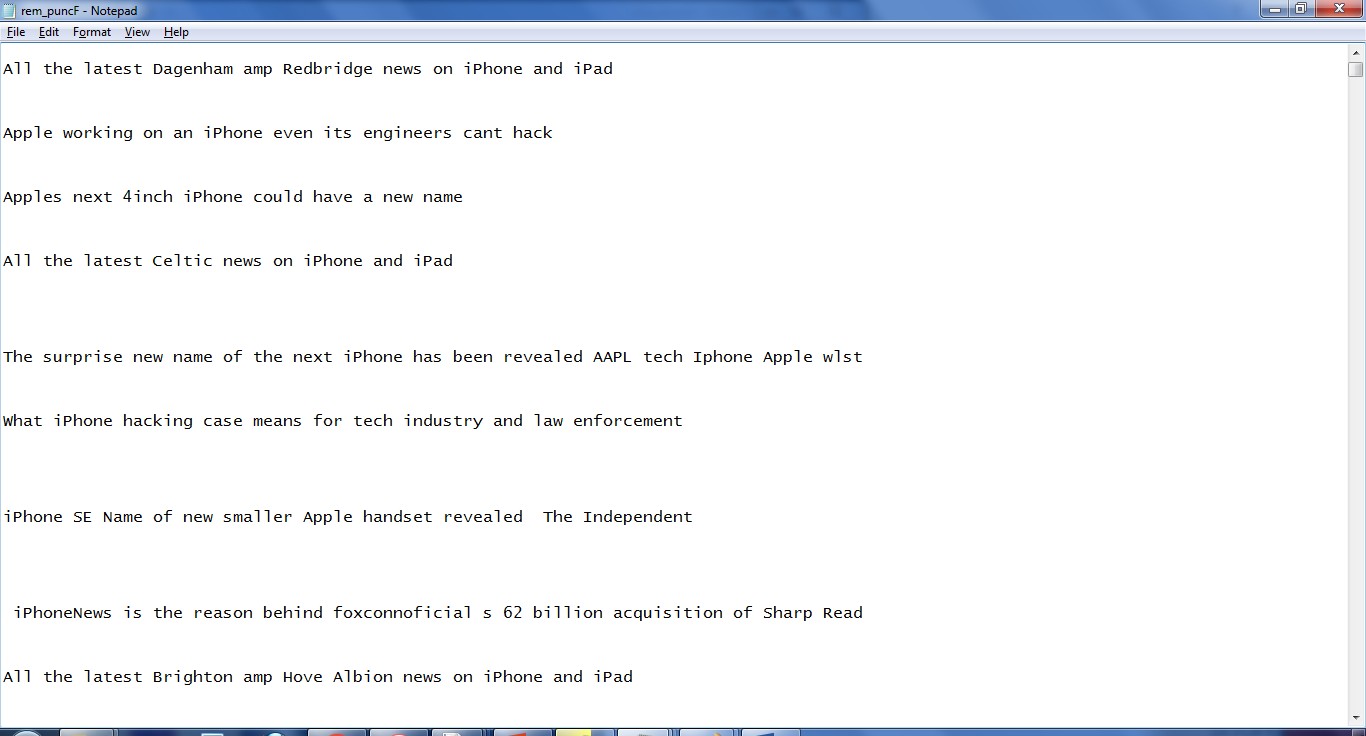
**Fig 6.1: Streaming of tweets from Twitter API**



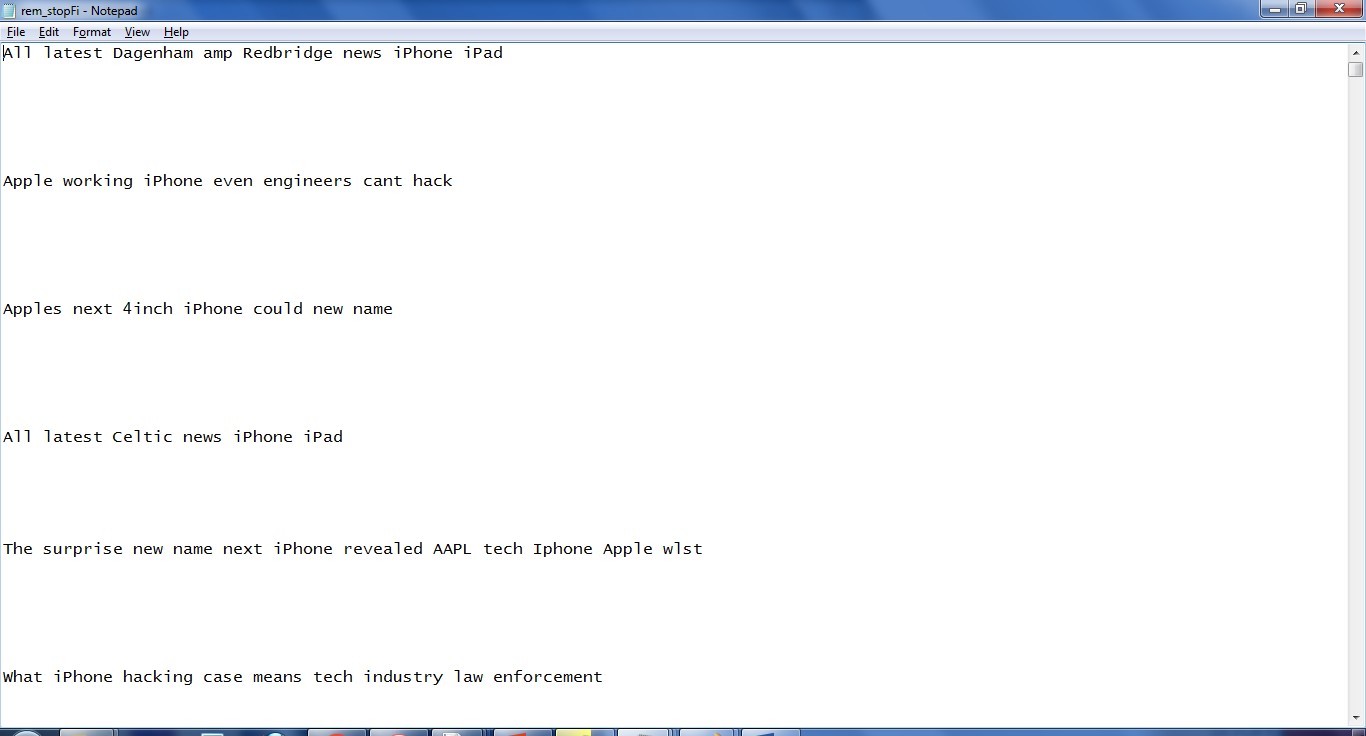
**Fig 6.2: Json Dump of streamed data**



**Fig 6.3: URL cleaned Data**



**Fig 6.4: Data free from Punctuations**



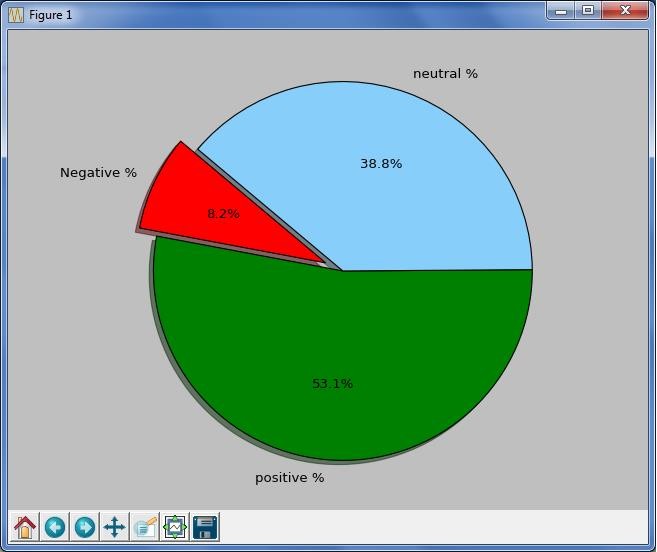
**Fig 6.5: Data free from Stopwords**



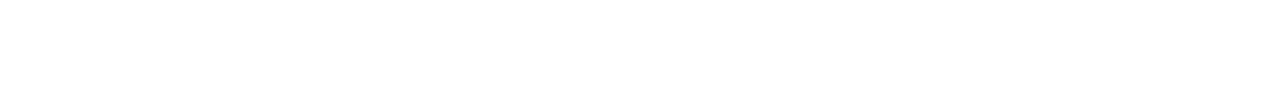
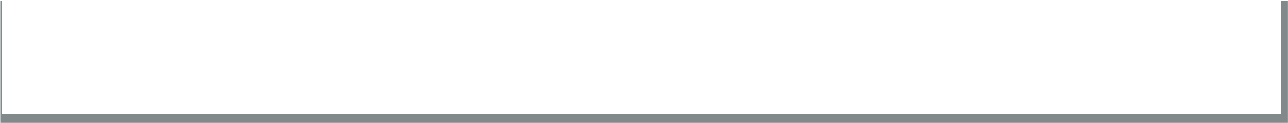
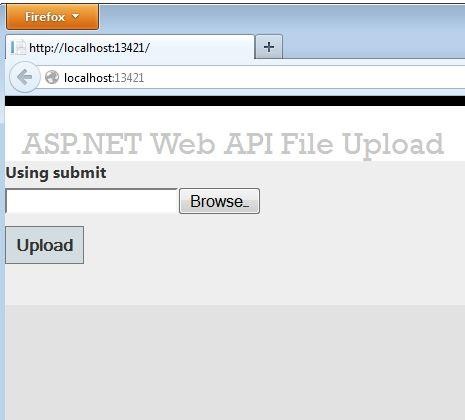
**Fig 6.6: Removal of excess space**



**Fig 6.7: Sentiment Analysis of Tweets using Naïve Bayes Classifier**

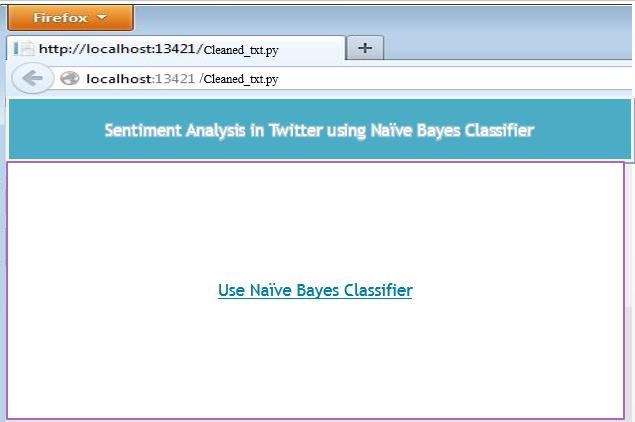


**Fig 6.9: Pie Chart representation of acquired results**

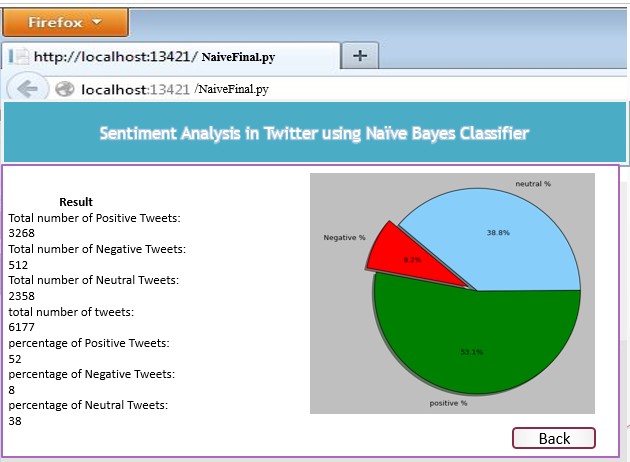


Sentiment Analysis in Twitter using Naïve Bayes Classifier

**Fig 6.10 uploading cleaned data file**



**Fig 6.11 Using of Naïve Bayes Classifier on uploaded file**



**Fig 6.12 Final Result in web browser**