REF:JKU/2/83/022

Jomo Kenyatta University of Agriculture and Technology (JKUAT)

School of Computing and Information Technology (SCIT)

Department of Computing

System Analysis Design And Implementation

Title: An Aspect-based Sentiment Analysis technique to predict product performance in promotional campaigns.

Student Name:KIRAGU PHILLIS WACERA Reg. No:CS282-0763/2011

Submission Date:27/08/2015

Supervisor 1:SYLVESTER KIPTOO

Supervisor 2:DR.AGNES MINDILA

Period: March 2015

Abstract

Textual information in the world can be broadly be categorized into two main types: facts and opinions. Facts are objective expressions about entities, events and their properties. Opinions are usually subjective expressions that describe people's sentiments, appraisals or feelings toward entities, events and their properties. Much of the existing research on textual information processing has been focused on mining and retrieval of factual information, e.g., information retrieval, Web search, text classification, text clustering and many other text mining and natural language processing tasks. Little work had been done on the processing of opinions until only recently. Yet, opinions are so important that whenever we need to make a decision we want to hear others opinions. This is not only true for individuals but also true for organizations.

This document discuss sentiment analysis, the field that is considered with opinion mining from user generated content. It describes the challenges, techniques and application of sentiment analysis. It will concentrate on aspect based analysis which is a sentiment analysis technique that analyses individual phrases in sentences in order to determine the orientation, that is, positive neutral and negative

The result should provide a prototype that is able to take sentiments and determine their orientation and then display the results to the interested parties, that is, promotional campaign managers.

The previous documents included my literature review on sentiment analysis my research methodology and my proposal.

JKUAT is ISO 9000:2008 certified
Setting Trend in Higher Education, Research and Innovation

Contents

1 INTRODUCTION

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. It is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes.(Liu 2012) This presentation will deal with the analysis and design of an aspect based sentiment analysis system.

2 SYSTEM ANALYSIS

2.1 Feasibility study

Economic

The system is economically feasible because it helps predict market performance hence reducing losses incurred due to poorly performing products. The resources needed to develop and use it are not expensive but it requires quite some amount of time but once its up and running its benefits surpasses the resources used. The system can be used to analyze any kind of words in English making it possible to be used by different people to analyze different texts.

Technical

All the resources needed to build the system are available, including the necessary skill set, software and hardware requirements rendering the system feasible technically.

Operation

Operation feasibility: It is concerned with how easy the system is to operate. As it is a desktop application with a graphical interface it is therefore easy to use For the efficient operation of the system, the user needs a general computer. The GUI is a simple window, so it do es not require any special skill to use it in addition a help menu will be provided. The system user will have efficient processing, easy and fast access to sentiment analysis results. The proposed system is therefore operationally feasible.

Social and Political

This system does not compromise any social or political cultures as reviews and tweets are usually available for public use and some websites even provide licenses to developers using their corpus data.

2.2 Requirement Specification

2.2.1 System Requirements

- Windows/linux operating system
- Python 2.7 or above
- NLTK Toolkit
- PyQT
- Twitter API(Tweep)
- Libraries that are dependencies python eg matplotlib, numpy, scikit learn etc

2.2.2 Functional Requirements

Retrieving Input

The software will receive two inputs: Python code and Python libraries, and Tweets. Python code and Python libraries. Tweets will be retrieved from the Twitter API to be tested as they stream live.

Real-Time Processing

The software will take input, process data, and display output in real-time. It will also plot graphs in real time for the output.

Sentiment Analysis

Sentiment analysis will be performed on the keywords within the Tweet to determine the overall mood of the Tweet relative to the topic. The sentiment analysis will provide a negative or positive numeric sentiment value.

Output

The software must output the polarity of the sentiments in form of positive, negative and neutral. and show a graphic representation.

2.2.3 Nonfunctional Requirements

Performance

The Twitter API will provide up-to-date information; limited only by the rate of Twitter input. Python will provide promptly analysis of the data using the various software packages available to it. The output should display the latest results at all times, and if it lags behind, the user should be notified. The application should be capable of operating in the background should the user wish to utilize other applications.

Reliability

The software will meet all of the functional requirements without any unexpected behavior. At no time should the output display incorrect or outdated information without alerting the user to potential errors. In this instance error message will be shown.

Availability

The software will be available at all times on the user's device desktop or laptop, as long as the device is in proper working order. The functionality of the software will depend on any external services such as internet access that are required. If those services are unavailable, the user should be alerted.

Security

The software should never disclose any personal information of Twitter users, and should collect no personal information from its own users. The use of passwords and API keys will ensure private use of the Twitter API. The programming will be performed on a password protected laptop and desktop to ensure maximum security.

Maintainability

The software should be written clearly and concisely. The code will be well documented. Particular care will be taken to design the software modularly to ensure that maintenance is easy.

Portability

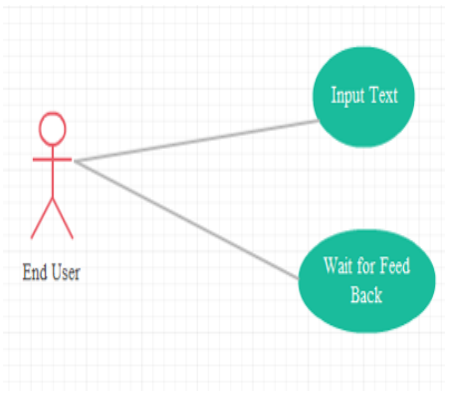
This software will be designed to run on any Windows operating system

2.2.4 User Requirements

The users of this system will require no skills as all they will need is to input the sentiment they want analyzed.

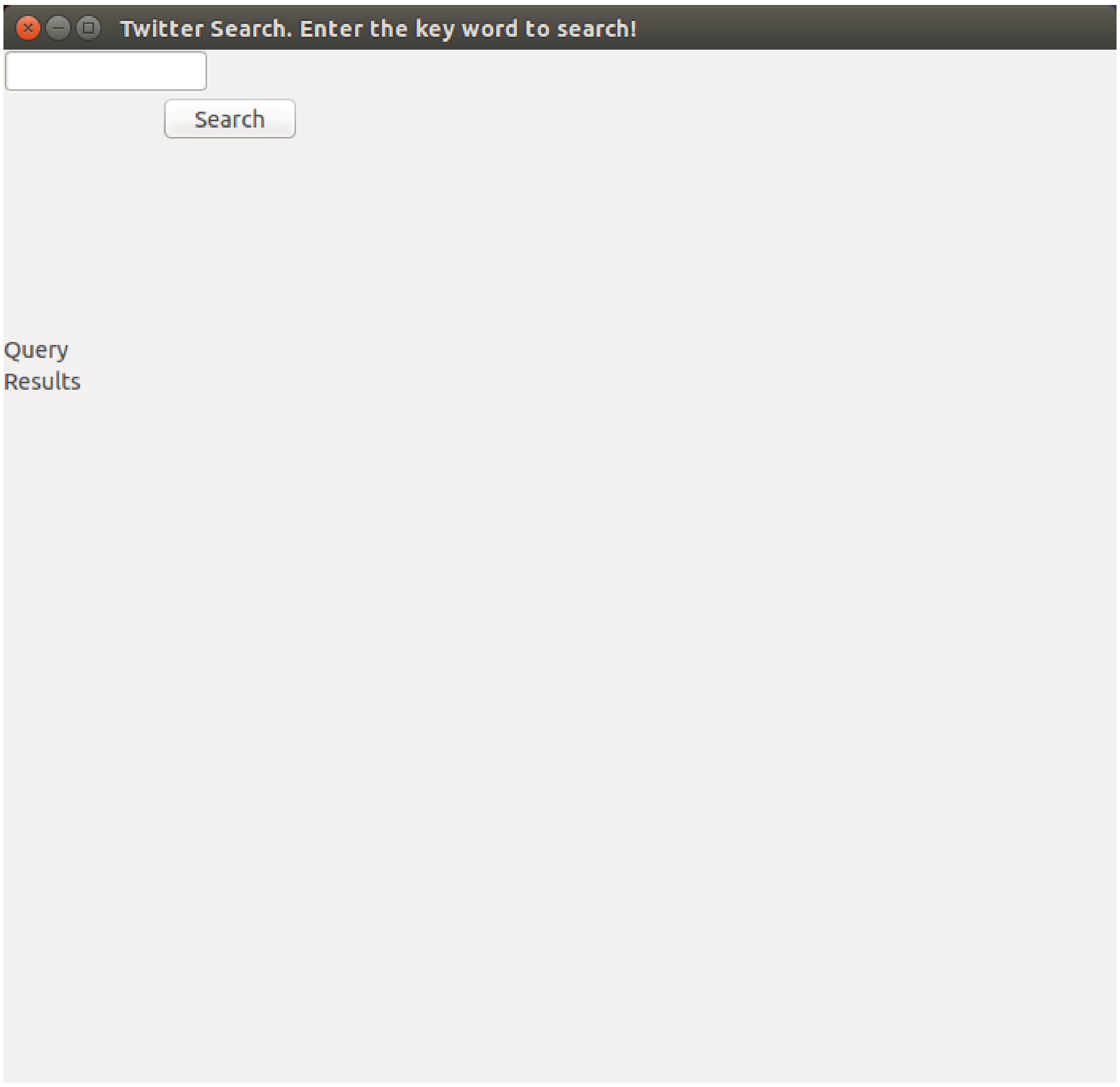
User Case Diagrams

The User As An Actor:



2.2.5 User Interface

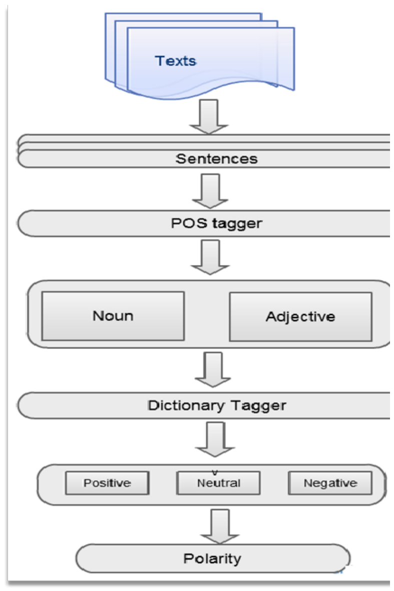
There is a simple user iterface that enables the user to search tweets with a particular key word.



3 SYSTEM DESIGN

3.1 Flow Chart Diagram

The flow chart explains how the system will be designed in order to accomplish the objectives of the system.



3.1.1 Process Design

The system will take texts as input and and pass it through a process called preprocessing. Before preprocessing it is imortant to determine the structure of the sentences.

The structure is as follows:

- Each text is a list of sentences
- Each sentence is a list of tokens
- Each token is a tuple of three elements: a word form (the exact word that appeared in the text), a word lemma (a generalized version of the word), and a list of associated tags

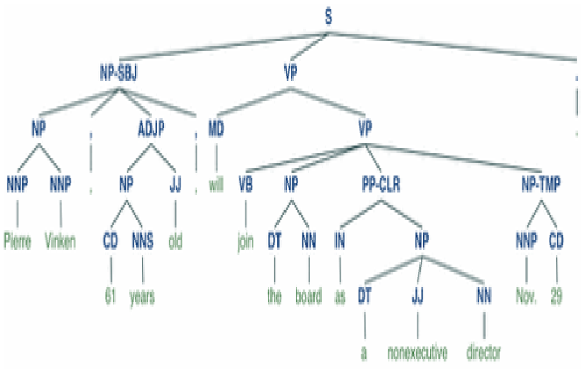
Next is to pass the POS tagged text through the dictionary taggers in order to determine their polarity. There are Several dictionary taggers:

- Positive
- Negative
- Incrementers
- Decrementers
- Inverters and Polarity Flips

The output at this point is the polarity of the text in question This is a lexicon based method.

3.1.2 NLTK(Natural Language Processing ToolKit)

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

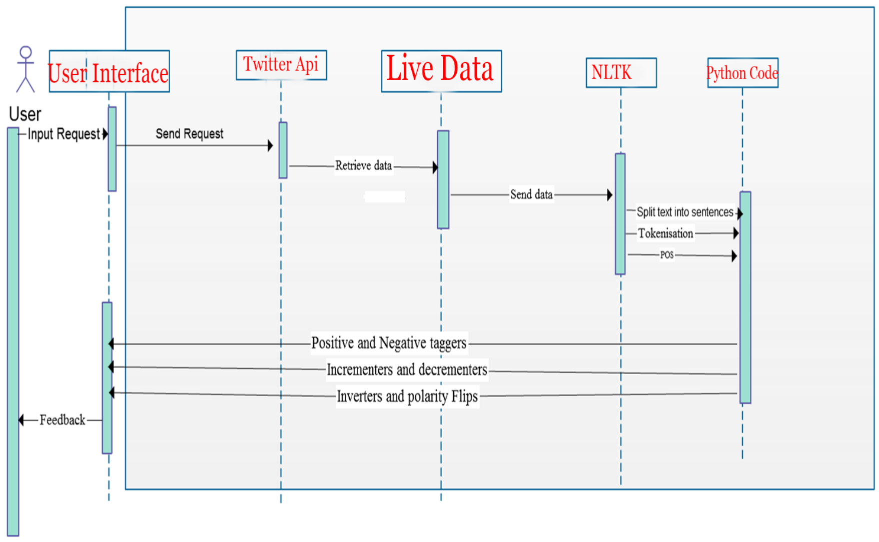


The pre-prossesing will be done using this tool kit. It will also be used to train the various classifiers that will be used in the system these will include:

- Naive Bayes
- MultinomialNB
- BernoulliNB
- Logistic Regression classifier
- SGD Classifier classifier
- Linear SVC classifier

These classifiers will be used to compare accuracy of the sentiment analysis model.

3.2 Sequence Diagram



The sequence diagram show how data will move from the twitter API after a user enters text through the processes until finally a user gets feed back.

3.3 Output

The output will be in form of graphs and charts that will display the polarity of tweets. They will show the numbers clearly such that one can tell whether the mood of the topic is positive , negative or neutral

3.4 Data set

Nltk contains a corpus full of data that is both positive and negative and will be used to train the model. The data used for the output will be retrieved using twitter API and should stream live.

3.5 Design considerations and assumptions

The system uses twitter API which limits the number of tweets that can be retrieved at a time. Only 1500 tweets can be retrieved at a time

Sometimes there are not enough tweets on a topic hence reducing the accuracy of the polarity test.

There is a lot of slang usage in twitter and some of those words do not exist in the dictionaries.

Also most people do not use correct grammar to tweet and the model is trained against English data.

4 SYSTEM IMPLEMENTATION

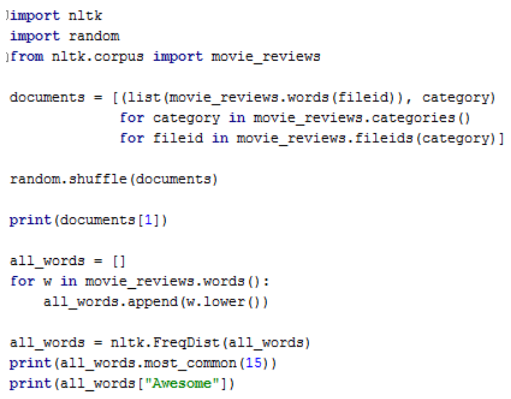
4.1 Platforms Used

The system has been implemented through python and using Linux OS, Ubuntu, It also incorporates multiple libraries and packages to enhance it they include:

- NLTK Toolkit
- Tweep API
- Scikit Learn
- Pickle
- Matplotlib
- Statistics
- PyQT(UI Design)

This document will explain how the implementation was carried out:

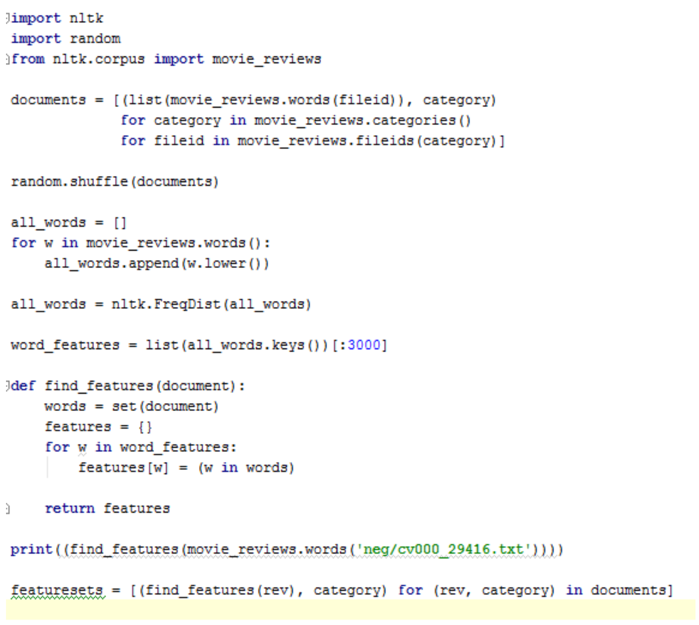
4.2 Text Classification



```python
import nltk
import random
from nltk.corpus import movie_reviews

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

print(documents[1])

all_words = []
for w in movie_reviews.words():
    all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)
print(all_words.most_common(15))
print(all_words["Awesome"])
```

This code snippet is an example of how the text is classified in NLTK, it will serve as a basis for the model and gives an output of the 15 most frequent words and also the number of times the word stupid has been used in the NLTK corpus Movie reviews which consist of 1000 positive and 1000 negative Reviews about movies.

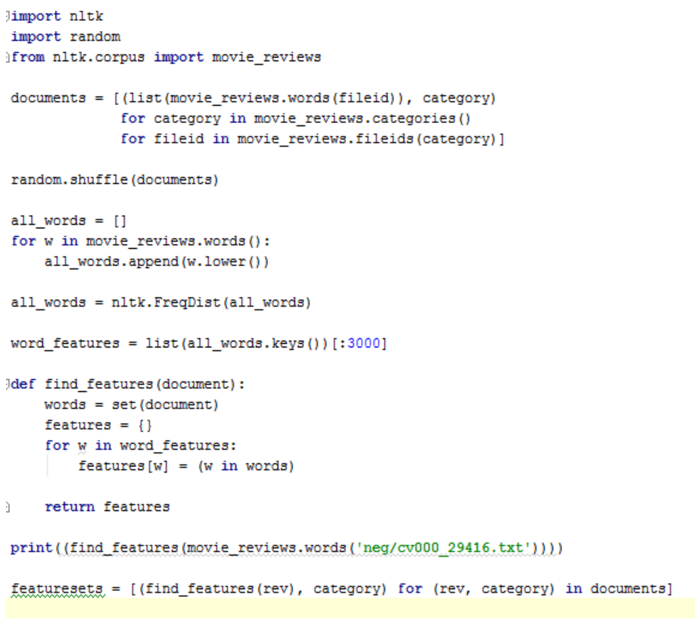
4.3 Words as Features



```
import nltk
import random
from nltk.corpus import movie_reviews

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = []
for w in movie_reviews.words():
    all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)

word_features = list(all_words.keys())[:3000]

def find_features(document):
    words = set(document)
    features = {}
    for w in word_features:
        features[w] = (w in words)

    return features

print((find_features(movie_reviews.words('neg/cv000_29416.txt'))))

featuresets = [(find_features(rev), category) for (rev, category) in documents]
```

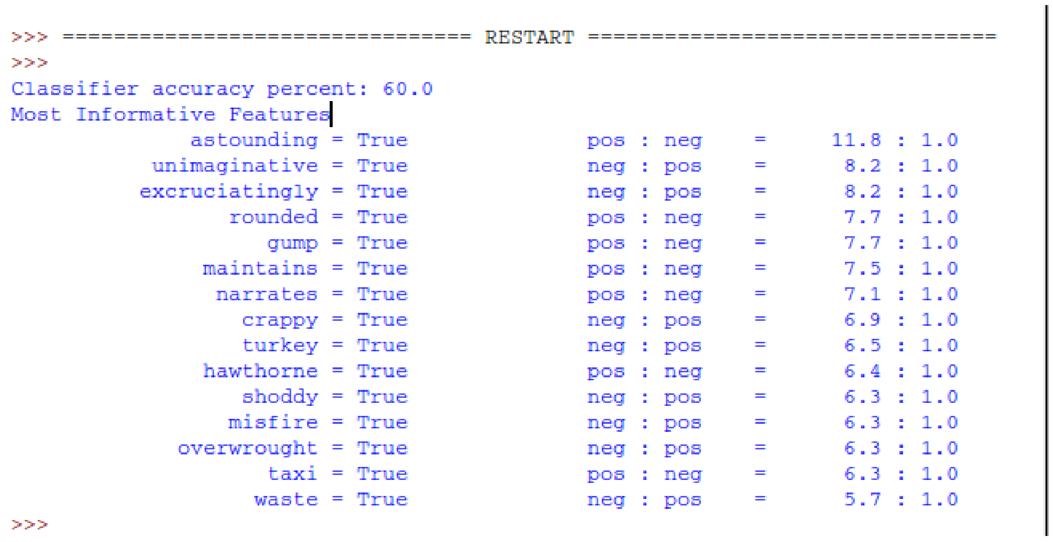
Here there is an introduction of a new variable, word-features, which contains the top 3000 most common words. A quick function that will find these top 3000 words in our positive and negative documents, marking their presence as either positive or negative is used.

4.4 Naive Bayes Classifier



```
import nltk
import random
from nltk.corpus import movie_reviews

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = []
for w in movie_reviews.words():
    all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)

word_features = list(all_words.keys())[:3000]

def find_features(document):
    words = set(document)
    features = {}
    for w in word_features:
        features[w] = (w in words)

    return features

print((find_features(movie_reviews.words('neg/cv000_29416.txt'))))

featuresets = [(find_features(rev), category) for (rev, category) in documents]
```

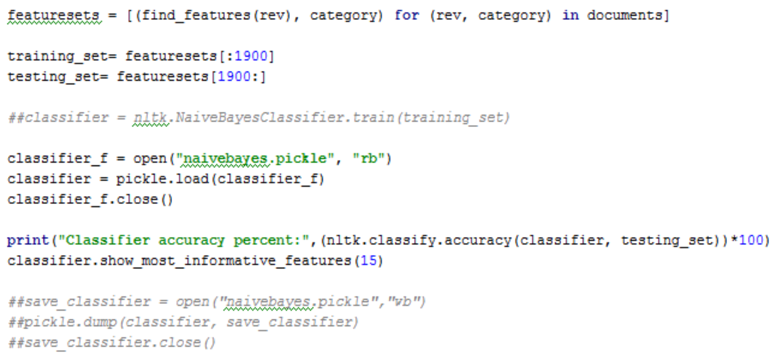
This code will train the model using naive bayes to tell us the polarity of features and then measure the accuracy of the classifier.



```
>>> ================================ RESTART ================================
>>>
Classifier accuracy percent: 60.0
Most Informative Features
              astounding = True              pos : neg    =     11.8 : 1.0
           unimaginative = True              neg : pos    =      8.2 : 1.0
          excruciatingly = True              neg : pos    =      8.2 : 1.0
                 rounded = True              pos : neg    =      7.7 : 1.0
                    gump = True              pos : neg    =      7.7 : 1.0
               maintains = True              pos : neg    =      7.5 : 1.0
                narrates = True              pos : neg    =      7.1 : 1.0
                  crappy = True              neg : pos    =      6.9 : 1.0
                  turkey = True              neg : pos    =      6.5 : 1.0
               hawthorne = True              pos : neg    =      6.4 : 1.0
                  shoddy = True              neg : pos    =      6.3 : 1.0
                 misfire = True              neg : pos    =      6.3 : 1.0
             overwrought = True              neg : pos    =      6.3 : 1.0
                    taxi = True              pos : neg    =      6.3 : 1.0
                   waste = True              neg : pos    =      5.7 : 1.0
>>>
```

4.5 Using Pickle to save documents

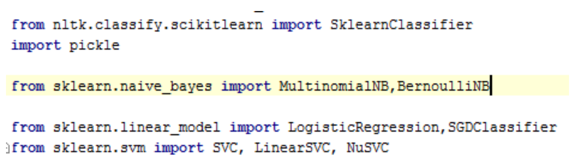
```
import pickle
```



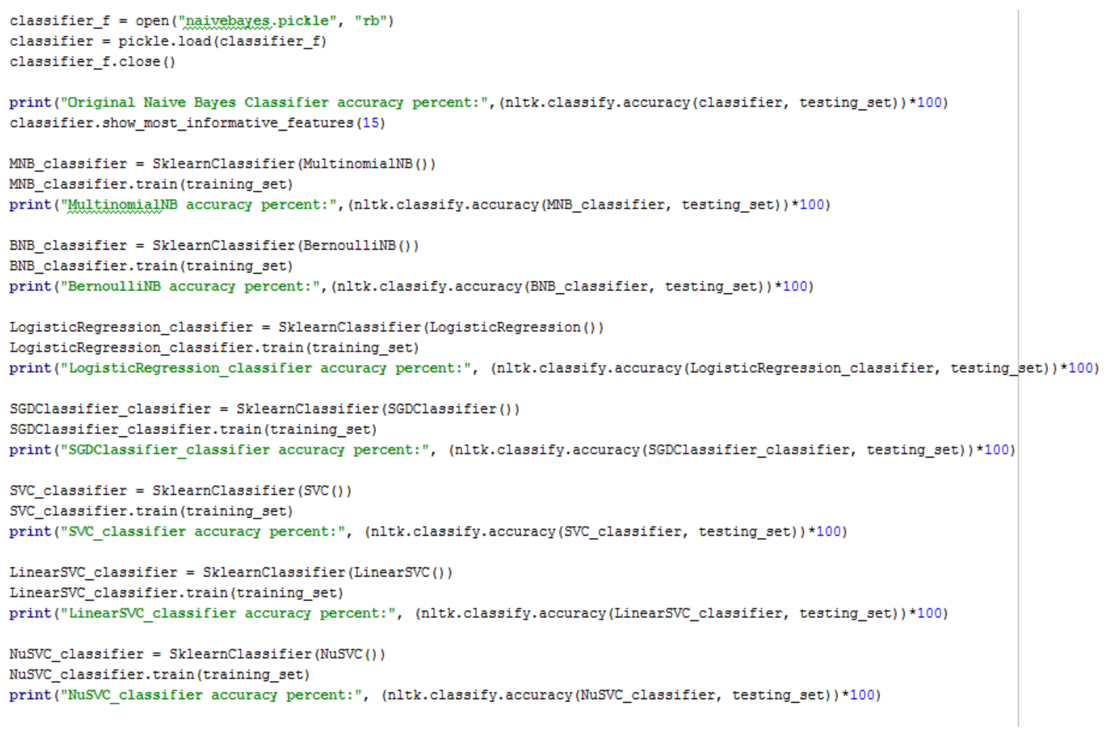
```
featuresets = [(find_features(rev), category) for (rev, category) in documents]

training_set= featuresets[:1900]
testing_set= featuresets[1900:]

##classifier = nltk.NaiveBayesClassifier.train(training_set)

classifier_f = open("naivebayes.pickle", "rb")
classifier = pickle.load(classifier_f)
classifier_f.close()

print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing_set))*100)
classifier.show_most_informative_features(15)

##save_classifier = open("naivebayes.pickle","wb")
##pickle.dump(classifier, save_classifier)
##save_classifier.close()
```

This code snippet is used to safe executed files, It is important because it saves the trained data, this helps to save time the next time you run the code. It is especially essential when using multiple classifiers. The next time you just need to load the already saved file.

4.6 Scikit Learn Classifiers

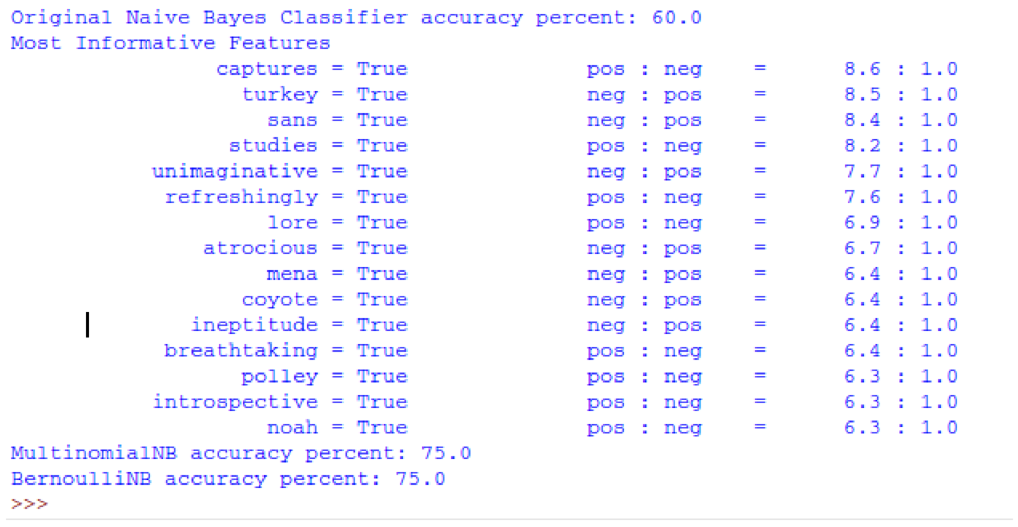


```
from nltk.classify.scikitlearn import SklearnClassifier
import pickle

from sklearn.naive_bayes import MultinomialNB,BernoulliNB

from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
```

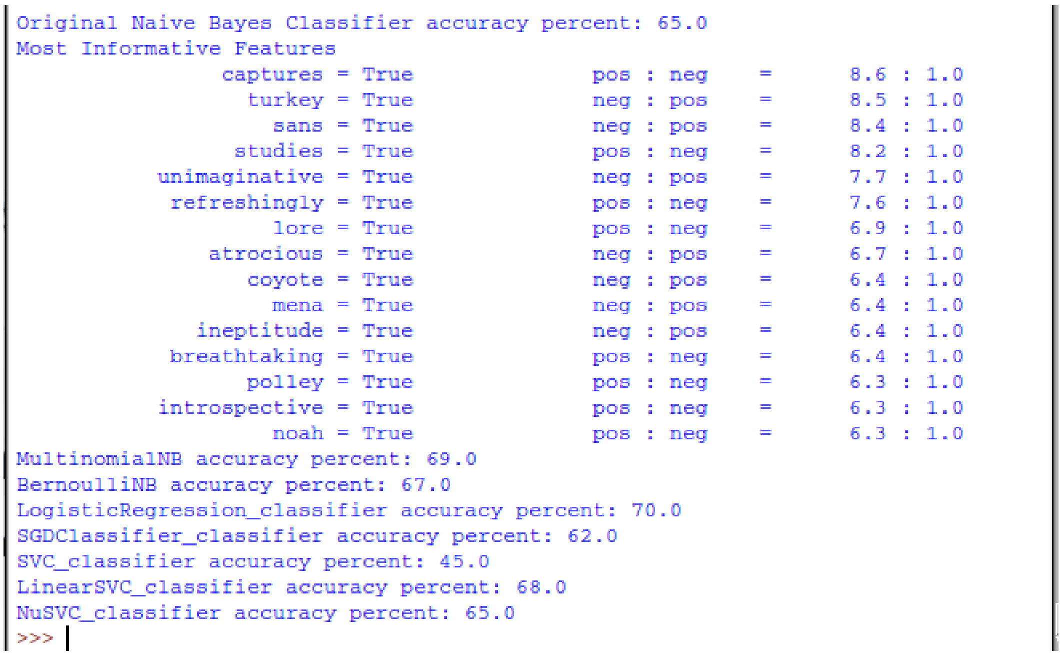


```
classifier_f = open("naivebayes.pickle", "rb")
classifier = pickle.load(classifier_f)
classifier_f.close()

print("Original Naive Bayes Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing_set))*100)
classifier.show_most_informative_features(15)

MNB_classifier = SklearnClassifier(MultinomialNB())
MNB_classifier.train(training_set)
print("MultinomialNB accuracy percent:",(nltk.classify.accuracy(MNB_classifier, testing_set))*100)

BNB_classifier = SklearnClassifier(BernoulliNB())
BNB_classifier.train(training_set)
print("BernoulliNB accuracy percent:",(nltk.classify.accuracy(BNB_classifier, testing_set))*100)

LogisticRegression_classifier = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier.train(training_set)
print("LogisticRegression_classifier accuracy percent:", (nltk.classify.accuracy(LogisticRegression_classifier, testing_set))*100)

SGDClassifier_classifier = SklearnClassifier(SGDClassifier())
SGDClassifier_classifier.train(training_set)
print("SGDClassifier_classifier accuracy percent:", (nltk.classify.accuracy(SGDClassifier_classifier, testing_set))*100)

SVC_classifier = SklearnClassifier(SVC())
SVC_classifier.train(training_set)
print("SVC_classifier accuracy percent:", (nltk.classify.accuracy(SVC_classifier, testing_set))*100)

LinearSVC_classifier = SklearnClassifier(LinearSVC())
LinearSVC_classifier.train(training_set)
print("LinearSVC_classifier accuracy percent:", (nltk.classify.accuracy(LinearSVC_classifier, testing_set))*100)

NuSVC_classifier = SklearnClassifier(NuSVC())
NuSVC_classifier.train(training_set)
print("NuSVC_classifier accuracy percent:", (nltk.classify.accuracy(NuSVC_classifier, testing_set))*100)
```

We include other classifiers from the scikit learn package so that we are able to compare the accuracy of each of them and determine which gives a better result.

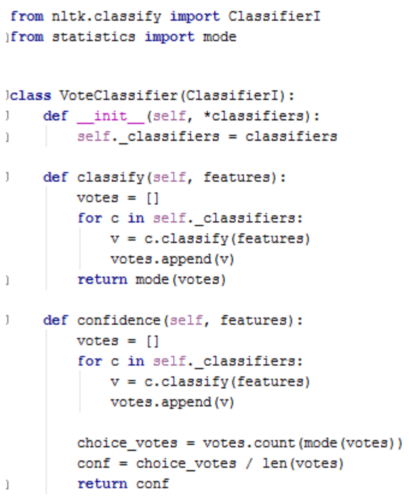


```
Original Naive Bayes Classifier accuracy percent: 60.0
Most Informative Features
                captures = True              pos : neg    =      8.6 : 1.0
                  turkey = True              neg : pos    =      8.5 : 1.0
                    sans = True              neg : pos    =      8.4 : 1.0
                 studies = True              pos : neg    =      8.2 : 1.0
           unimaginative = True              neg : pos    =      7.7 : 1.0
            refreshingly = True              pos : neg    =      7.6 : 1.0
                    lore = True              pos : neg    =      6.9 : 1.0
               atrocious = True              neg : pos    =      6.7 : 1.0
                    mena = True              neg : pos    =      6.4 : 1.0
                  coyote = True              neg : pos    =      6.4 : 1.0
              ineptitude = True              neg : pos    =      6.4 : 1.0
            breathtaking = True              pos : neg    =      6.4 : 1.0
                  polley = True              pos : neg    =      6.3 : 1.0
           introspective = True              pos : neg    =      6.3 : 1.0
                    noah = True              pos : neg    =      6.3 : 1.0
MultinomialNB accuracy percent: 75.0
BernoulliNB accuracy percent: 75.0
>>>
```

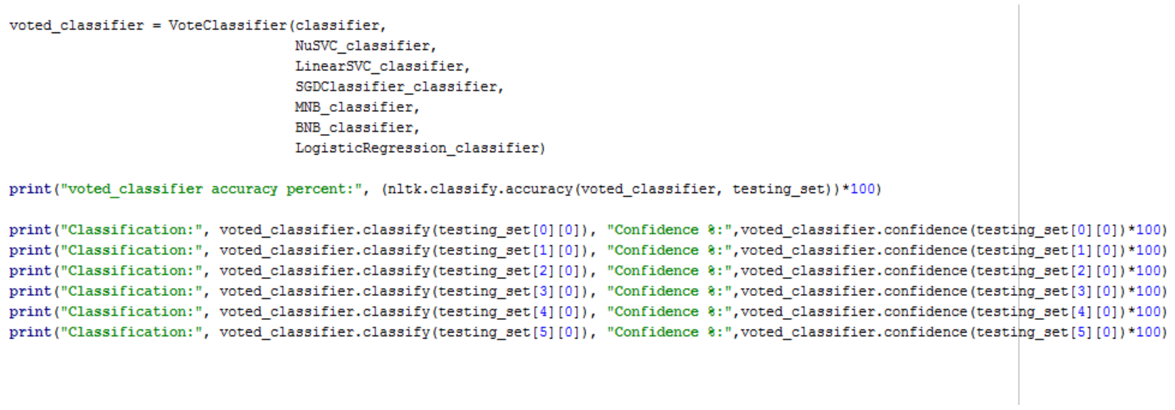


```
Original Naive Bayes Classifier accuracy percent: 65.0
Most Informative Features
                captures = True              pos : neg    =      8.6 : 1.0
                  turkey = True              neg : pos    =      8.5 : 1.0
                    sans = True              neg : pos    =      8.4 : 1.0
                 studies = True              pos : neg    =      8.2 : 1.0
           unimaginative = True              neg : pos    =      7.7 : 1.0
            refreshingly = True              pos : neg    =      7.6 : 1.0
                    lore = True              pos : neg    =      6.9 : 1.0
               atrocious = True              neg : pos    =      6.7 : 1.0
                  coyote = True              neg : pos    =      6.4 : 1.0
                    mena = True              neg : pos    =      6.4 : 1.0
              ineptitude = True              neg : pos    =      6.4 : 1.0
            breathtaking = True              pos : neg    =      6.4 : 1.0
                  polley = True              pos : neg    =      6.3 : 1.0
           introspective = True              pos : neg    =      6.3 : 1.0
                    noah = True              pos : neg    =      6.3 : 1.0
MultinomialNB accuracy percent: 69.0
BernoulliNB accuracy percent: 67.0
LogisticRegression_classifier accuracy percent: 70.0
SGDClassifier_classifier accuracy percent: 62.0
SVC_classifier accuracy percent: 45.0
LinearSVC_classifier accuracy percent: 68.0
NuSVC_classifier accuracy percent: 65.0
>>> 
```

4.7 Combining Classifiers

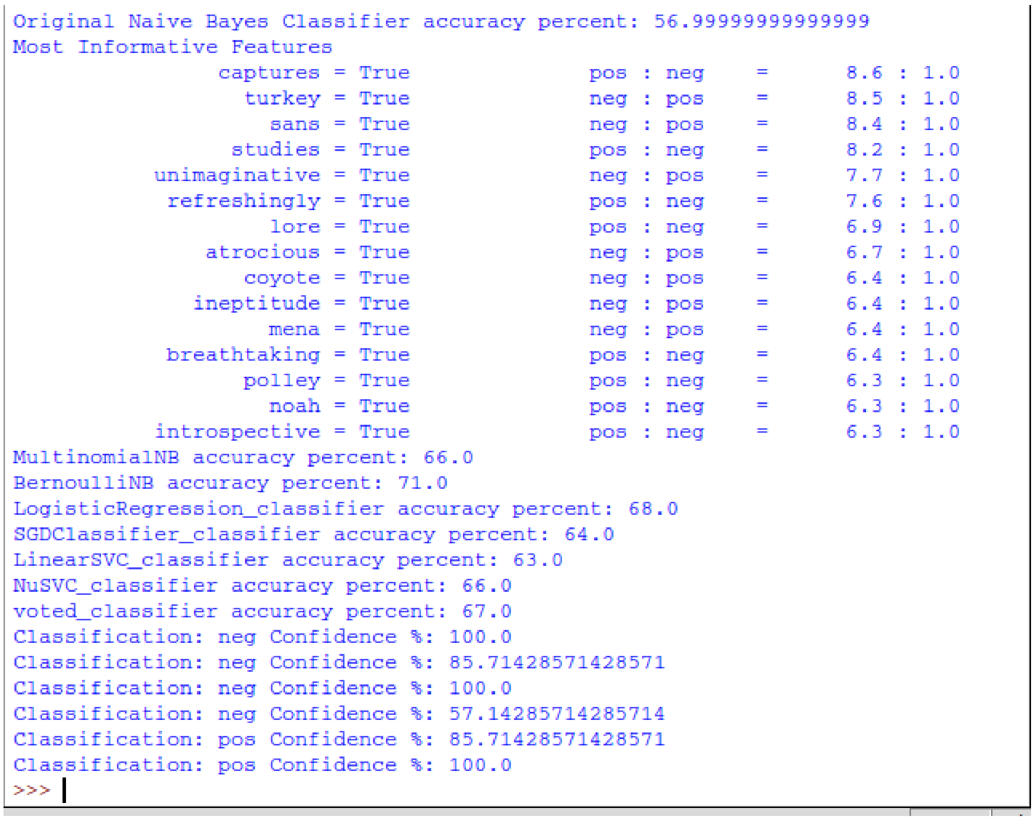


```
from nltk.classify import ClassifierI
from statistics import mode


class VoteClassifier(ClassifierI):
    def __init__(self, *classifiers):
        self._classifiers = classifiers

    def classify(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)
        return mode(votes)

    def confidence(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)

        choice_votes = votes.count(mode(votes))
        conf = choice_votes / len(votes)
        return conf
```



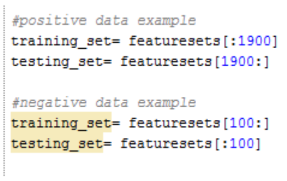
```
voted_classifier = VoteClassifier(classifier,
                                  NuSVC_classifier,
                                  LinearSVC_classifier,
                                  SGDClassifier_classifier,
                                  MNB_classifier,
                                  BNB_classifier,
                                  LogisticRegression_classifier)

print("voted_classifier accuracy percent:", (nltk.classify.accuracy(voted_classifier, testing_set))*100)

print("Classification:", voted_classifier.classify(testing_set[0][0]), "Confidence %:",voted_classifier.confidence(testing_set[0][0])*100)
print("Classification:", voted_classifier.classify(testing_set[1][0]), "Confidence %:",voted_classifier.confidence(testing_set[1][0])*100)
print("Classification:", voted_classifier.classify(testing_set[2][0]), "Confidence %:",voted_classifier.confidence(testing_set[2][0])*100)
print("Classification:", voted_classifier.classify(testing_set[3][0]), "Confidence %:",voted_classifier.confidence(testing_set[3][0])*100)
print("Classification:", voted_classifier.classify(testing_set[4][0]), "Confidence %:",voted_classifier.confidence(testing_set[4][0])*100)
print("Classification:", voted_classifier.classify(testing_set[5][0]), "Confidence %:",voted_classifier.confidence(testing_set[5][0])*100)
```

We combine the algorithms and use a vote classifier to determine the most frequently appearing accuracy score so as to determine the ideal accuracy. we also measure the confidence level.



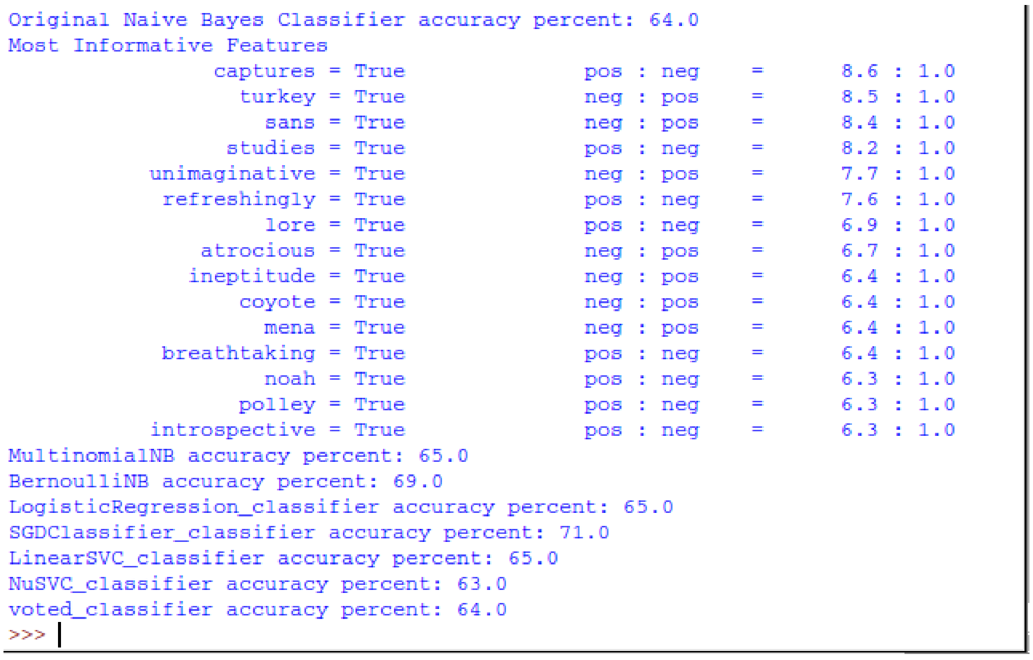
```
Original Naive Bayes Classifier accuracy percent: 56.99999999999999
Most Informative Features
                captures = True              pos : neg    =      8.6 : 1.0
                  turkey = True              neg : pos    =      8.5 : 1.0
                    sans = True              neg : pos    =      8.4 : 1.0
                 studies = True              pos : neg    =      8.2 : 1.0
           unimaginative = True              neg : pos    =      7.7 : 1.0
            refreshingly = True              pos : neg    =      7.6 : 1.0
                    lore = True              pos : neg    =      6.9 : 1.0
               atrocious = True              neg : pos    =      6.7 : 1.0
                  coyote = True              neg : pos    =      6.4 : 1.0
               ineptitude = True              neg : pos    =      6.4 : 1.0
                    mena = True              neg : pos    =      6.4 : 1.0
             breathtaking = True              pos : neg    =      6.4 : 1.0
                  polley = True              pos : neg    =      6.3 : 1.0
                    noah = True              pos : neg    =      6.3 : 1.0
            introspective = True              pos : neg    =      6.3 : 1.0
MultinomialNB accuracy percent: 66.0
BernoulliNB accuracy percent: 71.0
LogisticRegression_classifier accuracy percent: 68.0
SGDClassifier_classifier accuracy percent: 64.0
LinearSVC_classifier accuracy percent: 63.0
NuSVC_classifier accuracy percent: 66.0
voted_classifier accuracy percent: 67.0
Classification: neg Confidence %: 100.0
Classification: neg Confidence %: 85.71428571428571
Classification: neg Confidence %: 100.0
Classification: neg Confidence %: 57.14285714285714
Classification: pos Confidence %: 85.71428571428571
Classification: pos Confidence %: 100.0
>>> 
```

4.8 Investigating Bias



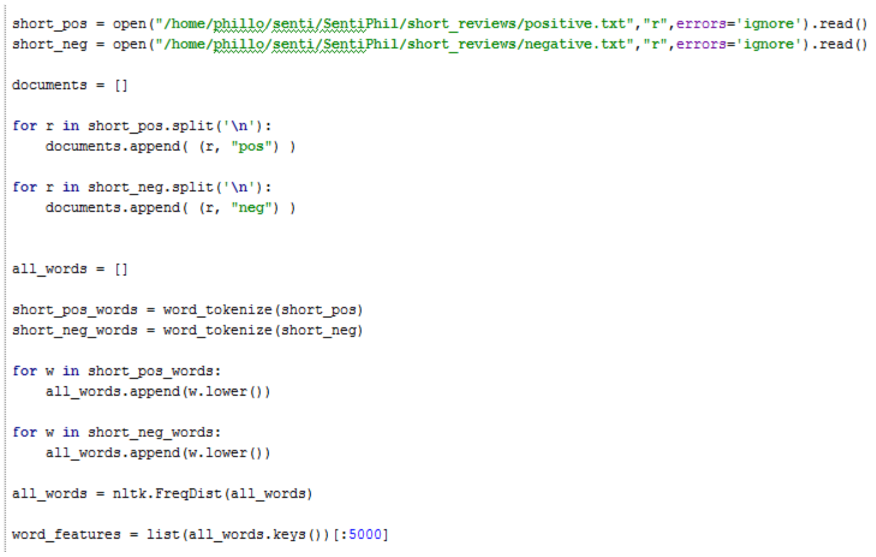
```
#positive data example
training_set= featuresets[:1900]
testing_set= featuresets[1900:]

#negative data example
training_set= featuresets[100:]
testing_set= featuresets[:100]
```

To investigate bias we remove the random feature and test for each positive and negative separately



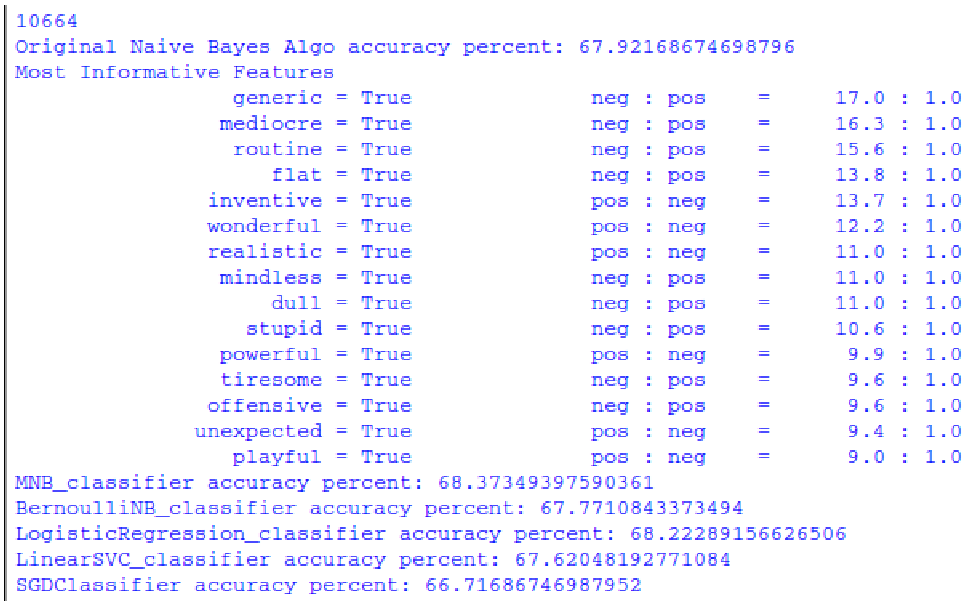
```
Original Naive Bayes Classifier accuracy percent: 64.0
Most Informative Features
                captures = True              pos : neg    =      8.6 : 1.0
                  turkey = True              neg : pos    =      8.5 : 1.0
                    sans = True              neg : pos    =      8.4 : 1.0
                 studies = True              pos : neg    =      8.2 : 1.0
           unimaginative = True              neg : pos    =      7.7 : 1.0
            refreshingly = True              pos : neg    =      7.6 : 1.0
                    lore = True              pos : neg    =      6.9 : 1.0
               atrocious = True              neg : pos    =      6.7 : 1.0
               ineptitude = True             neg : pos    =      6.4 : 1.0
                  coyote = True              neg : pos    =      6.4 : 1.0
                    mena = True              neg : pos    =      6.4 : 1.0
            breathtaking = True              pos : neg    =      6.4 : 1.0
                    noah = True              pos : neg    =      6.3 : 1.0
                  polley = True              pos : neg    =      6.3 : 1.0
           introspective = True              pos : neg    =      6.3 : 1.0
MultinomialNB accuracy percent: 65.0
BernoulliNB accuracy percent: 69.0
LogisticRegression_classifier accuracy percent: 65.0
SGDClassifier_classifier accuracy percent: 71.0
LinearSVC_classifier accuracy percent: 65.0
NuSVC_classifier accuracy percent: 63.0
voted_classifier accuracy percent: 64.0
>>>
```

4.9 Conducting better training



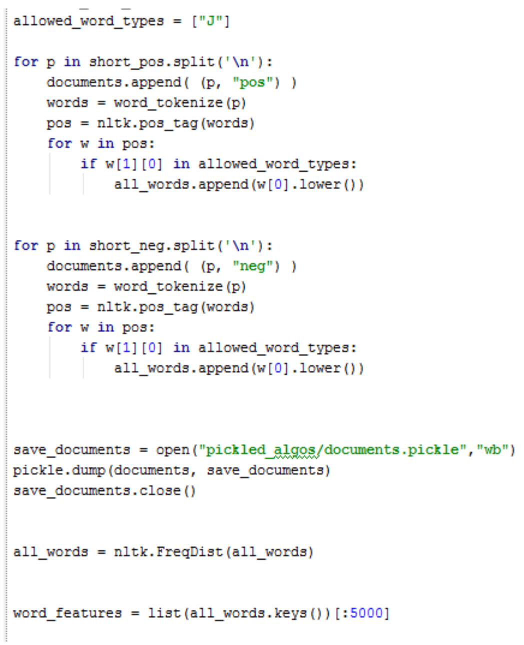
```
short_pos = open("/home/phillo/senti/SentiPhil/short_reviews/positive.txt","r",errors='ignore').read()
short_neg = open("/home/phillo/senti/SentiPhil/short_reviews/negative.txt","r",errors='ignore').read()

documents = []

for r in short_pos.split('\n'):
    documents.append( (r, "pos") )

for r in short_neg.split('\n'):
    documents.append( (r, "neg") )


all_words = []

short_pos_words = word_tokenize(short_pos)
short_neg_words = word_tokenize(short_neg)

for w in short_pos_words:
    all_words.append(w.lower())

for w in short_neg_words:
    all_words.append(w.lower())

all_words = nltk.FreqDist(all_words)

word_features = list(all_words.keys())[:5000]
```

Since we are looking to test the polarity of tweets which are at most 140 characters it would be better to train the model against shorter reviews so we use the text files to train the model and measure the accuracy. In this case we removed some of the classifiers that were inaccurate.

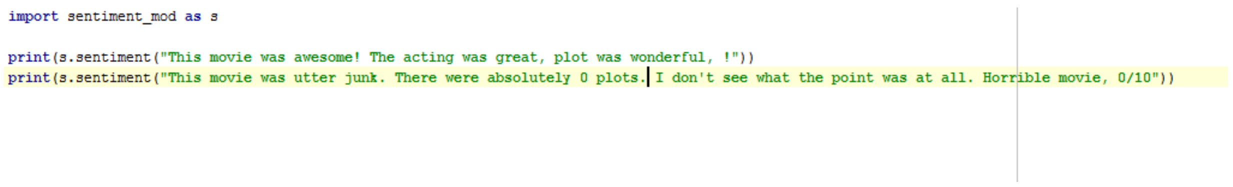


```
10664
Original Naive Bayes Algo accuracy percent: 67.92168674698796
Most Informative Features
                 generic = True              neg : pos    =     17.0 : 1.0
                mediocre = True              neg : pos    =     16.3 : 1.0
                 routine = True              neg : pos    =     15.6 : 1.0
                    flat = True              neg : pos    =     13.8 : 1.0
               inventive = True              pos : neg    =     13.7 : 1.0
               wonderful = True              pos : neg    =     12.2 : 1.0
               realistic = True              pos : neg    =     11.0 : 1.0
                mindless = True              neg : pos    =     11.0 : 1.0
                    dull = True              neg : pos    =     11.0 : 1.0
                  stupid = True              neg : pos    =     10.6 : 1.0
                powerful = True              pos : neg    =      9.9 : 1.0
                tiresome = True              neg : pos    =      9.6 : 1.0
               offensive = True              neg : pos    =      9.6 : 1.0
              unexpected = True              pos : neg    =      9.4 : 1.0
                 playful = True              pos : neg    =      9.0 : 1.0
MNB_classifier accuracy percent: 68.37349397590361
BernoulliNB_classifier accuracy percent: 67.7710843373494
LogisticRegression_classifier accuracy percent: 68.22289156626506
LinearSVC_classifier accuracy percent: 67.62048192771084
SGDClassifier accuracy percent: 66.71686746987952
```

4.10 Creating a sentiment analysis module

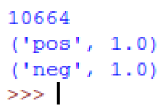


```
allowed_word_types = ["J"]

for p in short_pos.split('\n'):
    documents.append( (p, "pos") )
    words = word_tokenize(p)
    pos = nltk.pos_tag(words)
    for w in pos:
        if w[1][0] in allowed_word_types:
            all_words.append(w[0].lower())


for p in short_neg.split('\n'):
    documents.append( (p, "neg") )
    words = word_tokenize(p)
    pos = nltk.pos_tag(words)
    for w in pos:
        if w[1][0] in allowed_word_types:
            all_words.append(w[0].lower())



save_documents = open("pickled_algos/documents.pickle","wb")
pickle.dump(documents, save_documents)
save_documents.close()


all_words = nltk.FreqDist(all_words)


word_features = list(all_words.keys())[:5000]
```



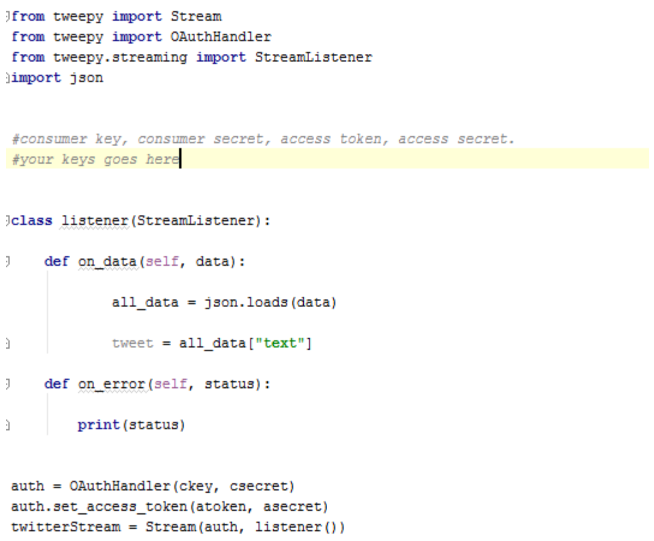
```
import sentiment_mod as s

print(s.sentiment("This movie was awesome! The acting was great, plot was wonderful, !"))
print(s.sentiment("This movie was utter junk. There were absolutely 0 plots. I don't see what the point was at all. Horrible movie, 0/10"))
```

We use the previous code to build the sentiment analysis module. This time we will use the pos-tag(part of speech) function in the code so as to return the parts of speech specified. we also use more nltk features such as word tokenize.



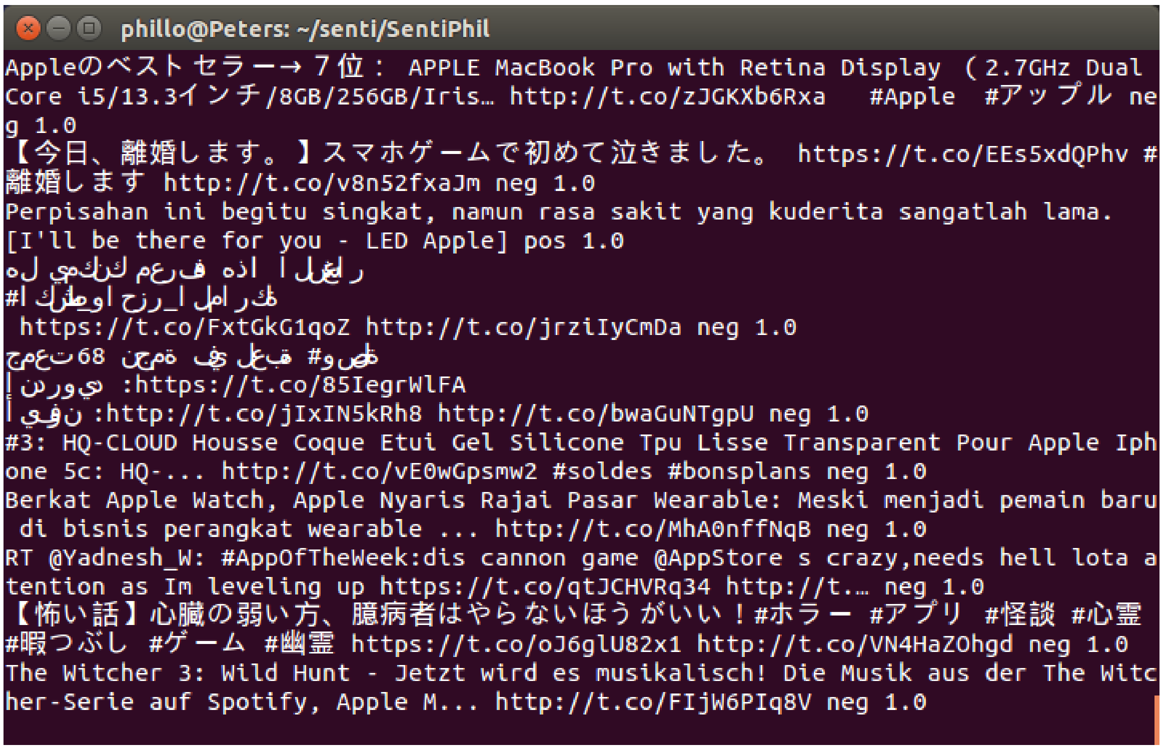
```
10664
('pos', 1.0)
('neg', 1.0)
>>> 
```

4.11 Getting Twitter Data



```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import json


#consumer key, consumer secret, access token, access secret.
#your keys goes here


class listener(StreamListener):

    def on_data(self, data):

            all_data = json.loads(data)

            tweet = all_data["text"]

    def on_error(self, status):

        print(status)


auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
twitterStream = Stream(auth, listener())
```

This is the code we will use to stream data live from the twitter API. You need secret keys for you to access twitter to stream tweets.

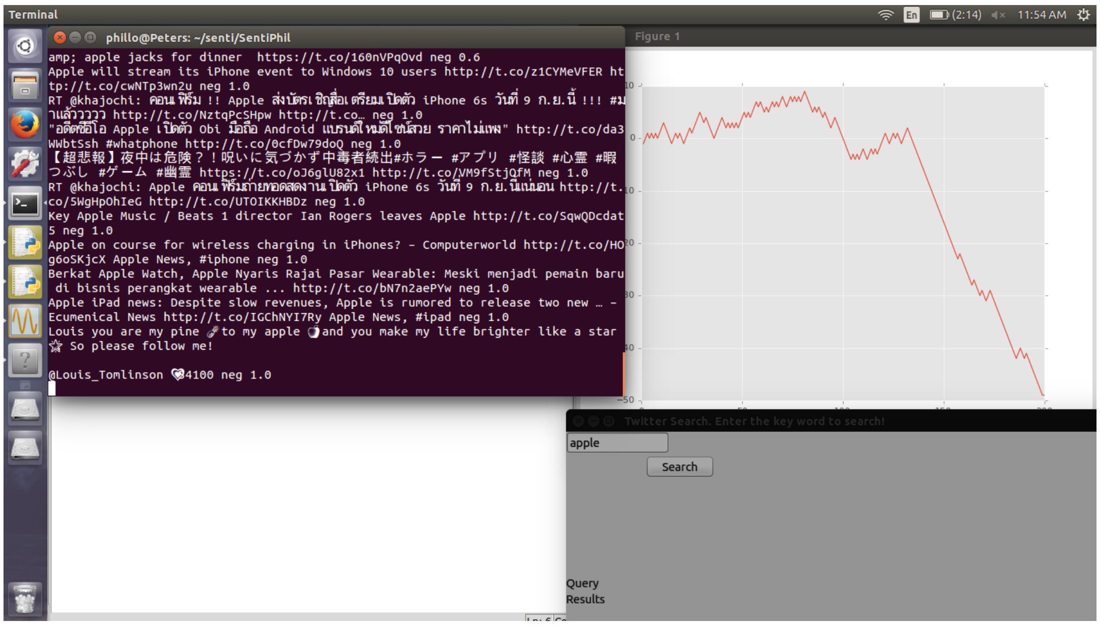


```
Appleのベストセラー→７位： APPLE MacBook Pro with Retina Display （2.7GHz Dual
Core i5/13.3インチ/8GB/256GB/Iris… http://t.co/zJGKXb6Rxa  #Apple  #アップル ne
g 1.0
【今日、離婚します。】スマホゲームで初めて泣きました。 https://t.co/EEs5xdQPhv #
離婚します http://t.co/v8n52fxaJm neg 1.0
Perpisahan ini begitu singkat, namun rasa sakit yang kuderita sangatlah lama.
[I'll be there for you - LED Apple] pos 1.0
هل يمكنك فرعم اذه ا لغيلر
#ا كلطوا حزر_ا لماركك
 https://t.co/FxtGkG1qoZ http://t.co/jrziIyCmDa neg 1.0
جمعت68 نجمة في علبة #وصلت
إ نيروي :https://t.co/85IegrWlFA
أ نفي :http://t.co/jIxIN5kRh8 http://t.co/bwaGuNTgpU neg 1.0
#3: HQ-CLOUD Housse Coque Etui Gel Silicone Tpu Lisse Transparent Pour Apple Iph
one 5c: HQ-... http://t.co/vE0wGpsmw2 #soldes #bonsplans neg 1.0
Berkat Apple Watch, Apple Nyaris Rajai Pasar Wearable: Meski menjadi pemain baru
 di bisnis perangkat wearable ... http://t.co/MhA0nffNqB neg 1.0
RT @Yadnesh_W: #AppOfTheWeek:dis cannon game @AppStore s crazy,needs hell lota a
tention as Im leveling up https://t.co/qtJCHVRq34 http://t.… neg 1.0
【怖い話】心臓の弱い方、臆病者はやらないほうがいい！#ホラー #アプリ #怪談 #心霊
#暇つぶし #ゲーム #幽霊 https://t.co/oJ6glU82x1 http://t.co/VN4HaZOhgd neg 1.0
The Witcher 3: Wild Hunt - Jetzt wird es musikalisch! Die Musik aus der The Witc
her-Serie auf Spotify, Apple M... http://t.co/FIjW6PIq8V neg 1.0
```

4.12 Twitter Sentiment analysis

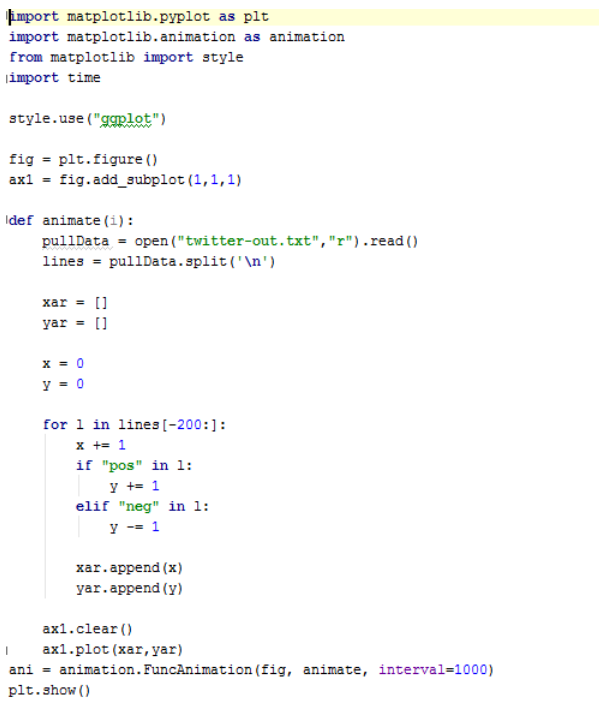
```
class listener(StreamListener):

    def on_data(self, data):

            all_data = json.loads(data)

            tweet = all_data["text"]
            sentiment_value, confidence = s.sentiment(tweet)
            print(tweet, sentiment_value, confidence)

            if confidence*100 >= 80:

                    output = open("twitter-out.txt","a")
                    output.write(sentiment_value)
                    output.write('\n')
                    output.close()

            return True

    def on_error(self, status):

        print(status)


auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
twitterStream = Stream(auth, listener())
```

Now we combine our module with the twitter data, so that we can perform the analysis. The following screen shows the output after running the program.


Terminal
phillo@Peters: ~/senti/SentiPhil
Figure 1
11:54 AM
Twitter Search. Enter the key word to search!
apple
Search
Query
Results

```
amp; apple jacks for dinner  https://t.co/160nVPqOvd neg 0.6
Apple will stream its iPhone event to Windows 10 users http://t.co/z1CYMeVFER ht
tp://t.co/cwNTp3wn2u neg 1.0
RT @khajochi: คอนเฟิร์ม !! Apple ส่งบัตรเชิญสื่อเตรียมเปิดตัว iPhone 6s วันที่ 9 ก.ย.นี้ !!! #ม
าแล้วววววว http://t.co/NztqPcSHpw http://t.co… neg 1.0
"อดีตซีอีโอ Apple เปิดตัว Obi มือถือ Android แบรนด์ใหม่ดีไซน์สวย ราคาไม่แพง" http://t.co/da3
WWbtSsh #whatphone http://t.co/0cfDw79doQ neg 1.0
【超悲報】夜中は危険？！呪いに気づかず中毒者続出#ホラー #アプリ #怪談 #心霊 #暇
つぶし #ゲーム #幽霊 https://t.co/oJ6glU82x1 http://t.co/VM9fStjQfM neg 1.0
RT @khajochi: Apple คอนเฟิร์มถ่ายทอดสดงานเปิดตัว iPhone 6s วันที่ 9 ก.ย.นี้แน่นอน http://t.
co/5WgHpOhIeG http://t.co/UTOIKKHBDz neg 1.0
Key Apple Music / Beats 1 director Ian Rogers leaves Apple http://t.co/SqwQDcdat
5 neg 1.0
Apple on course for wireless charging in iPhones? - Computerworld http://t.co/HO
g6oSKjcX Apple News, #iphone neg 1.0
Berkat Apple Watch, Apple Nyaris Rajai Pasar Wearable: Meski menjadi pemain baru
 di bisnis perangkat wearable ... http://t.co/bN7n2aePYw neg 1.0
Apple iPad news: Despite slow revenues, Apple is rumored to release two new … -
Ecumenical News http://t.co/IGChNYI7Ry Apple News, #ipad neg 1.0
Louis you are my pine 🍍to my apple 🍎and you make my life brighter like a star
⭐ So please follow me!

@Louis_Tomlinson 💝4100 neg 1.0
```

4.13 Graphing the output



```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from matplotlib import style
import time

style.use("ggplot")

fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)

def animate(i):
    pullData = open("twitter-out.txt","r").read()
    lines = pullData.split('\n')

    xar = []
    yar = []

    x = 0
    y = 0

    for l in lines[-200:]:
        x += 1
        if "pos" in l:
            y += 1
        elif "neg" in l:
            y -= 1

        xar.append(x)
        yar.append(y)

    ax1.clear()
    ax1.plot(xar,yar)
ani = animation.FuncAnimation(fig, animate, interval=1000)
plt.show()
```

This is the final part where we output the results of the twitter sentiment analysis in a graph using matplotlib

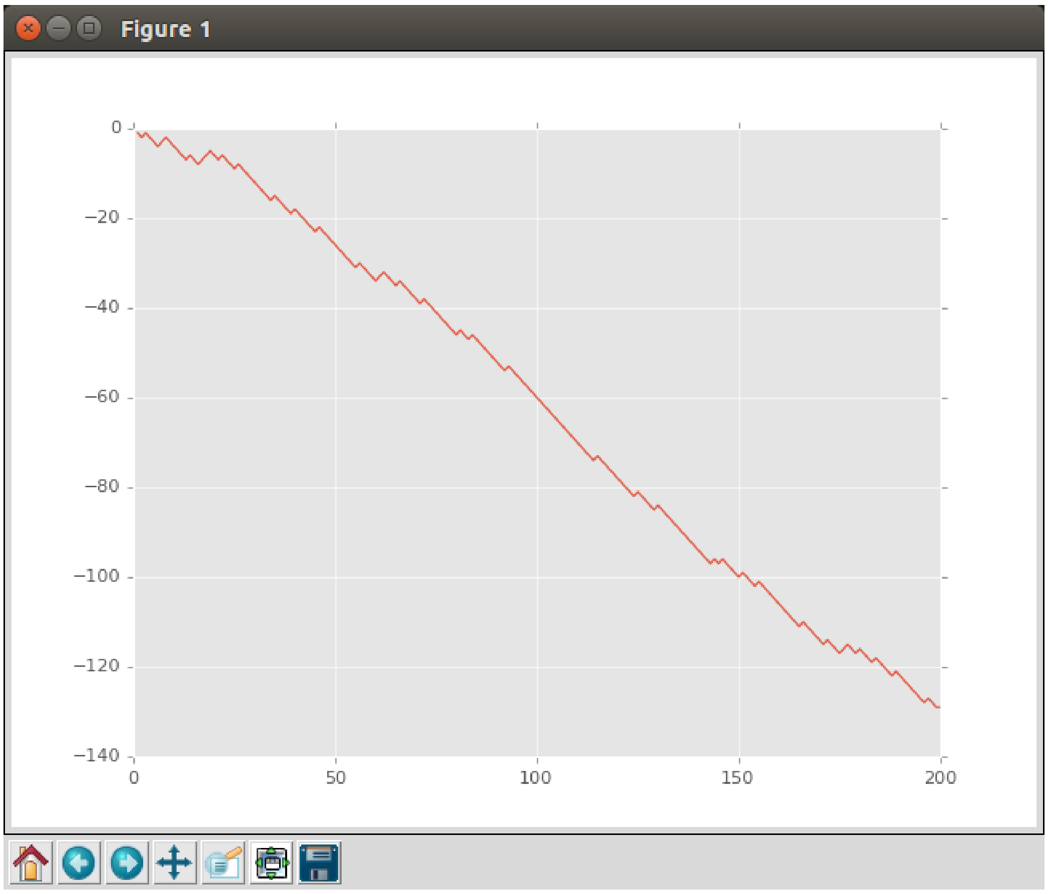
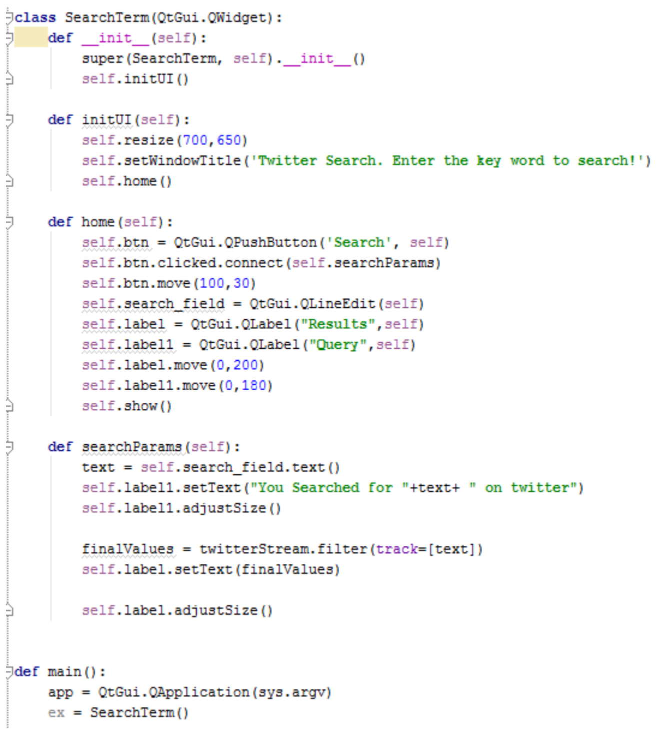


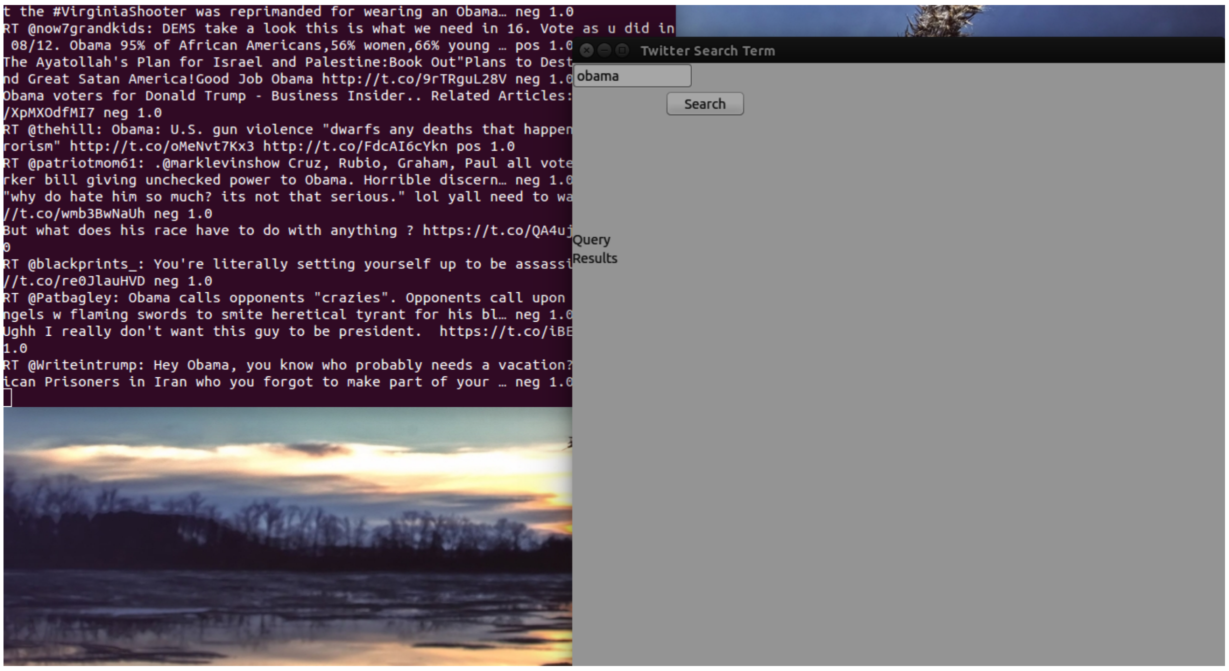
Figure 1
0
−20
−40
−60
−80
−100
−120
−140
0
50
100
150
200

4.14 User Interface



```python
class SearchTerm(QtGui.QWidget):
    def __init__(self):
        super(SearchTerm, self).__init__()
        self.initUI()

    def initUI(self):
        self.resize(700,650)
        self.setWindowTitle('Twitter Search. Enter the key word to search!')
        self.home()

    def home(self):
        self.btn = QtGui.QPushButton('Search', self)
        self.btn.clicked.connect(self.searchParams)
        self.btn.move(100,30)
        self.search_field = QtGui.QLineEdit(self)
        self.label = QtGui.QLabel("Results",self)
        self.label1 = QtGui.QLabel("Query",self)
        self.label.move(0,200)
        self.label1.move(0,180)
        self.show()

    def searchParams(self):
        text = self.search_field.text()
        self.label1.setText("You Searched for "+text+ " on twitter")
        self.label1.adjustSize()

        finalValues = twitterStream.filter(track=[text])
        self.label.setText(finalValues)

        self.label.adjustSize()


def main():
    app = QtGui.QApplication(sys.argv)
    ex = SearchTerm()
```

The user interface is simple and it provides the user with a search box to enter the keyword , the search button calls the twitter analysis method stream which in turn returns the tweets in real time. as the sentiment analysis module determines whether a tweet is positive or negative, as the graph changes in real time also. IT uses the PyQT GUI design.



```
t the #VirginiaShooter was reprimanded for wearing an Obama… neg 1.0
RT @now7grandkids: DEMS take a look this is what we need in 16. Vote as u did in
 08/12. Obama 95% of African Americans,56% women,66% young … pos 1.0
The Ayatollah's Plan for Israel and Palestine:Book Out"Plans to Dest
nd Great Satan America!Good Job Obama http://t.co/9rTRguL28V neg 1.0
Obama voters for Donald Trump - Business Insider.. Related Articles:
/XpMXOdfMI7 neg 1.0
RT @thehill: Obama: U.S. gun violence "dwarfs any deaths that happen
rorism" http://t.co/oMeNvt7Kx3 http://t.co/FdcAI6cYkn pos 1.0
RT @patriotmom61: .@marklevinshow Cruz, Rubio, Graham, Paul all vote
rker bill giving unchecked power to Obama. Horrible discern… neg 1.0
"why do hate him so much? its not that serious." lol yall need to wa
//t.co/wmb3BwNaUh neg 1.0
But what does his race have to do with anything ? https://t.co/QA4uj
0
RT @blackprints_: You're literally setting yourself up to be assassi
//t.co/re0JlauHVD neg 1.0
RT @Patbagley: Obama calls opponents "crazies". Opponents call upon
ngels w flaming swords to smite heretical tyrant for his bl… neg 1.0
Ughh I really don't want this guy to be president.  https://t.co/iBE
1.0
RT @Writeintrump: Hey Obama, you know who probably needs a vacation?
ican Prisoners in Iran who you forgot to make part of your … neg 1.0
```

5 CONCLUSION

The main challenge in training a sentiment analysis module is that there is no one time that any classifier will give a 100 percent accuracy. However any accuracy above 60 is good enough to get a good analysis.

Sentiment classification is a very important part of natural language processing and is very useful in determining the performance of a product. The system is able to search any keyword that a user might think of and if there tweets about it they will be streamead live and will be analyzed.

References

[1] *http://www.nltk.org/.*

[2] *https://docs.python.org/2/tutorial/.*

[3] Bird, Steven, Edward Loper and Ewan Klein *Natural Language Processing with Python.*, O'Reilly Media Inc, 2009.

[4] Liu, b. *Sentiment analysis and opinion mining.*, Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers, 2012.

[5] Turney, P. *Thumbs up or thumbs down?*,Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the Association for Computational Linguistics, 2002.

[6] Pang, b. and Lee, L. *a sentiment education: sentiment analysis using subjectivity summarization based on minimum cuts.*, In Proceedings of the Association for Computational Linguistics, 2004.

[7] Narayanan, r., Liu, b. and Chaudhary, a. *Sentiment analysis of conditional sentences.*, In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (Singapore), association for Computational Linguistics, 2009.