Sentiment analysis on twitter

Report submitted for the partial fulfillment of the requirements for the degree of Bachelor of Technology in
Information Technology

Submitted by

Name and Roll Number

Avijit Pal (IT2014/052)
11700214024

Argha Ghosh (IT2014/056)
11700214016

Bivuti Kumar (IT2014/061)
11700214083

Under the Guidance of Mr. **Amit Khan (Assistant Professor(IT),RCCIIT)**

RCC Institute of Information Technology
Canal South Road, Beliaghata, Kolkata – 700 015
[Affiliated to West Bengal University of Technology]

Acknowledgement

We would like to express our sincere gratitude to Mr. Amit khan of the department of Information Technology, whose role as project guide was invaluable for the project. We are extremely thankful 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 we convey our gratitude to all the teachers for providing us the technical skill that will always remain as our asset and to all non-teaching staff for the gracious hospitality they offered us.

Place: RCCIIT, Kolkata

Date: 14.05.2018

Avijit Pal

Argha Ghosh

Bivuti Kumar

Department of Information Technology
RCCIIT, Beliaghata,
Kolkata – 700 015,
West Bengal, India

Approval

This is to certify that the project report entitled "**Sentiment analysis on twitter"** prepared under my supervision by Avijit Pal (IT2014/052), Argha Ghosh (IT2014/056), Bivuti Kumar (IT2014/061) ., be accepted in partial fulfillment for the degree of Bachelor of Technology in Information Technology.

It is to be understood that by this approval, the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn thereof, but approves the report only.

……………………………………….. … ………………………………………

Dr.Abhjit Das.
HOD,IT
RCC Institute of Information Technology

Mr.Amit khan
Assistant Professor
RCC Institute of Information Technology

INDEX:

Introduction:

In the past few years, there has been a huge growth in the use of microblogging platforms such as Twitter. Spurred by that growth, companies and media organizations are increasingly seeking ways to mine Twitter for information about what people think and feel about their products and services. Companies such as Twitratr (twitrratr.com), tweetfeel (www.tweetfeel.com), and Social Mention (www.socialmention.com) are just a few who advertise Twitter sentiment analysis as one of their services.

While there has been a fair amount of research on how sentiments are expressed in genres such as online reviews and news articles, how sentiments are expressed given the informal language and message-length constraints of microblogging has been much less studied. Features such as automatic part-of-speech tags and resources such as sentiment lexicons have proved useful for sentiment analysis in other domains, but will they also prove useful for sentiment analysis in Twitter? In this paper, we begin to investigate this question.

Another challenge of microblogging is the incredible breadth of topic that is covered. It is not an exaggeration to say that people tweet about anything and everything. Therefore, to be able to build systems to mine Twitter sentiment about any given topic, we need a method for quickly identifying data that can be used for training. In this paper, we explore one method for building such data: using Twitter hashtags (e.g., #bestfeeling, #epicfail, #news) to identify positive, negative, and neutral tweets to use for training threeway sentiment classifiers.

The online medium has become a significant way for people to express their opinions and with social media, there is an abundance of opinion information available. Using sentiment analysis, the polarity of opinions can be found, such as positive, negative, or neutral by analyzing the text of the opinion. Sentiment analysis has been useful for companies to get their customer's opinions on their products predicting outcomes of elections , and getting opinions from movie reviews. The information gained from sentiment analysis is useful for companies making future decisions.
Many traditional approaches in sentiment analysis uses the bag of words method. The bag of words technique does not consider language morphology, and it could incorrectly classify two phrases of having the same meaning because it could have the same bag of words . The relationship between the collection of words is considered instead of the relationship between individual words . When determining the overall sentiment, the sentiment of each word is determined and combined using a function . Bag of words also ignores word order, which leads to phrases with negation in them to be incorrectly classified. Other techniques discussed in sentiment analysis include Naive Bayes, Maximum Entropy, and Support Vector Machines. In the Literature Survey section, approaches used for sentiment analysis and text classification are summarized.

Sentiment analysis refers to the broad area of natural language processing which deals with the computational study of opinions, sentiments and emotions expressed in text. Sentiment Analysis (SA) or Opinion Mining (OM) aims at learning people's opinions, attitudes and emotions towards an entity. The entity can represent individuals, events or topics. An immense amount of research has been performed in the area of sentiment analysis. But most of them focused on classifying formal and larger pieces of text data like reviews. With the wide popularity of social networking and microblogging websites and an immense amount of data available from these resources, research projects on sentiment analysis have witnessed a gradual domain shift. The past few years have witnessed a huge growth in the use of microblogging platforms. Popular microblogging websites like Twitter have evolved to become a source of varied information. This diversity in the information owes to such microblogs being elevated as platforms where people post real time messages about their opinions on a wide variety of topics, discuss current affairs and share their experience on products and services they use in daily life. Stimulated by the growth of microblogging platforms, organizations are exploring ways to mine Twitter for information about how people are responding to their products and services. A fair amount of research has been carried out on how sentiments are expressed in formal text patterns such as product or movie reviews and news articles, but how sentiments are expressed given the informal language and message-length constraints of microblogging has been less explored.

Twitter is an innovative microblogging service aired in 2006 with currently more than 550 million users 1. The user created status messages are termed tweets by this service. The public timeline of twitter service displays tweets of all users worldwide and is an extensive source of real-time information. The original concept behind microblogging was to provide personal status updates. But the current scenario surprisingly witnesses tweets covering everything under the world, ranging from current political affairs to personal experiences. Movie reviews, travel experiences, current events etc. add to the list. Tweets (and microblogs in general) are different from reviews in their basic structure. While reviews are characterized by formal text patterns and are summarized thoughts of authors, tweets are more casual and restricted to 140 characters of text. Tweets offer companies an additional avenue to gather feedback. Sentiment analysis to research products, movie reviews etc. aid customers in decision making before making a purchase or planning for a movie. Enterprises find this area useful to research public opinion of their company and products, or to analyze customer satisfaction. Organizations utilize this information to gather feedback about newly released products which supplements in improving further design. Different approaches which include machine learning(ML) techniques, sentiment lexicons, hybrid approaches etc. have been proved useful for sentiment analysis on formal texts. But their effectiveness for extracting sentiment in microblogging data will have to be explored. A careful investigation of tweets reveals that the 140 character length text restricts the vocabulary which imparts the sentiment. The hyperlinks often present in these tweets in turn restrict the vocabulary size. The varied domains discussed would surely impose hurdles for training. The frequency of misspellings and slang words in tweets (microblogs in general) is much higher than in other language resources which is another hurdle that needs to be overcome. On the other way around the tremendous volume of data available from microblogging websites on varied domains are incomparable with other data resources available. Microblogging language is characterized by expressive punctuations which convey a lot of sentiments. Bold lettered phrases, exclamations, question marks, quoted text etc. leave scope for sentiment extraction. The proposed work attempts a novel approach on twitter data by aggregating an adapted polarity lexicon which has learnt from product reviews of the domains under consideration, the tweet specific features and unigrams to build a classifier model using machine learning techniques.

Problem Definition:

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

Literature Survey:

Sentiment analysis is a growing area of Natural Language Processing 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 sentence-level 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.

Just in the past year 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). Other researchers have begun to explore the use of part-of-speech features but results remain mixed. Features common to microblogging (e.g., emoticons) are also common, but there has been little investigation into the usefulness of existing sentiment resources developed on non-microblogging data.

Researchers have also begun to investigate various ways of automatically collecting training data. Several researchers rely 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 use hashtags for creating training data, but they limit their experiments to sentiment/non-sentiment classification, rather than 3-way polarity classification, as we do.
We use WEKA and apply the following Machine Learning algorithms for this second classification to arrive at the best result:

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

In here we only use Naïve Bayes Classifiers to retrieve the Tweets from Twitter Data.

- **Naive Bayes Classification:**

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

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

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

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

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

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

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

$c = \text{argmax } P(c|d) \qquad \text{where } c \in C$

ThisideaofBayesianinferencehasbeenknownsincetheworkofBayes(1763),Bayesian inference and was first applied to text classification by Mosteller and Wallace (1964). The intuitionofBayesianclassificationistouseBayes'ruletotransformEq.6.1intoother probabilities that have some useful properties. Bayes' rule is presented in Eq. 2; it gives us a way to break down any conditional probability $P(x|y)$ into three other probabilities:

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

The algoritham of sentiment analysis using Naive Bayes Classification:

function BOOTSTRAP(x,b) returns p-value(x)

Calculate δ(x)

for i = 1to b do

for j = 1to n do # Draw a bootstrap sample x∗(i) of size n

Select a member of x at random and add it to x∗(i)

Calculate δ(x∗(i))

For each x∗(i)

s←s + 1ifδ(x∗(i)) > 2δ(x)

p-value(x)≈ s b

return p-value(x)

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

• Textcategorization,inwhichanentiretextisassignedaclassfromafiniteset, comprises such tasks as sentiment analysis, spam detection, email classification, and authorship attribution.

• Sentimentanalysisclassifiesatextasreflectingthepositiveornegativeorientation (sentiment) that a writer expresses toward some object.

• Naive Bayes is a generative model that make the bag of words assumption (positiondoesn’tmatter)andtheconditionalindependenceassumption(words are conditionally independent of each other given the class)

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

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

From textblob import TextBlob

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

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

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

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

The lexicon it refers to is in en-sentiment.xml, an XML document that includes the following four entries for the word “great”.

<word Form=”great” cornetto svnset id=”n_a-525317” wordnet id=”a-01123879” pos=”JJ” sense=”very good” polanty=”1.0” subjectivity=”1.0” intensity=”1.0” confidence=”0.9” />

<word Form=”great” wordnet id=”a-011238818” pos=”JJ” sense=”of major significance or importance” polanty=”1.0” subjectivity=”1.0” intensity=”1.0” confidence=”0.9” />

<word Form=”great” wordnet id=”a-01123883” pos=”JJ” sense=”relativity large in size or number or extent” polanty=”0.4” subjectivity=”0.2” intensity=”1.0” confidence=”0.9” />

<word Form=”great” wordnet id=”a-01677433” pos=”JJ” sense=”remarkable or out of the ordinary in degree or magnitude or effect” polanty=”0.8” subjectivity=”0.8” intensity=”1.0” confidence=”0.9” />

In addition to the polarity, subjectivity, and intensity mentioned in the comment above, there's also “confidence”, but I don't see this being used anywhere. In the case of “great” here it's all the same part of speech (JJ, adjective), and the senses are themselves natural language and not used. To simplify for readability:

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

When calculating sentiment for a single word, TextBlob uses a sophisticated technique known to

Mathematicians as “averaging”.

TextBlob(“great").sentiment

Sentiment(polarity=0.8, subjectivity=0.75)

At this point we might feel as if we're touring a sausage factory. That feeling isn't going to go away, but remember how delicious sausage is! Even if there isn't a lot of magic here, the results can be useful—and you certainly can't beat it for convenience.

TextBlob doesn't not handle negation, and that ain't nothing!

TextBlob("not great").sentiment

Sentiment(polarity=-0.4, subjectivity=0.75)

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

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

Word	Polarity	Subjectivity	Intensity
Very	0.2	0.3	1.3

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

TextBlob("very great").sentiment

Sentiment(polarity=1.0, subjectivity=0.9750000000000001)

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

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

TextBlob("not very great").sentiment

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

polarity=−0.5·11.3·0.8≈−0.31polarity=−0.5·11.3·0.8≈−0.31

subjectivity=11.3·0.75≈0.58subjectivity=11.3·0.75≈0.58

TextBlob will ignore one-letter words in its sentiment phrases, which means things like this will work just the same way:

TextBlob("not a very great").sentiment

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

And TextBlob will ignore words it doesn't know anything about:

TextBlob("not a very great calculation").sentiment

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

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

And while I'm being a little critical, and such a system of coded rules is in some ways the antithesis of machine learning, it is still a pretty neat system and I think I'd be hard-pressed to code up a better such solution.

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.

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.

- **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.

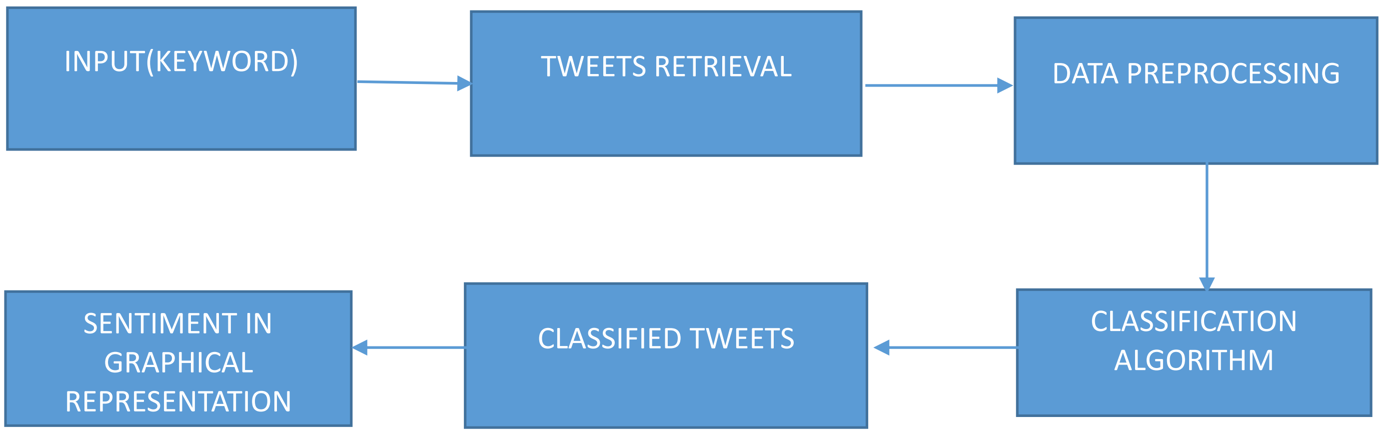
Software Quality Attributes:

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

Input(KeyWord):

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

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

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

Since we wanted to increase the generality of our data, we acquired it in portions at different points of time instead of acquiring all of it at one go. If we used the latter approach then the generality of the tweets might have been compromised since a significant portion of the tweets would be referring to some certain trending topic and would thus have more or less of the same general mood or sentiment. This phenomenon has been observed when we were going through our sample of acquired tweets. For example the sample acquired near Christmas and New Year's had a significant portion of tweets referring to these joyous events and were thus of a generally positive sentiment. Sampling our data in portions at different points in time would thus try to minimize this problem. Thus forth, we acquired data at four different points which would be 17th of December 2015, 29th of December 2015, 19th of January 2016 and 8th of February 2016.

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

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

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

Tweets Retrieval:

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

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

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

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

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

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

Data Preprocessing:

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

Tokenization:

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

Normalization:

For the normalization process, the presence of abbreviations within a tweet is noted and then abbreviations are replaced by their actual meaning (e.g., *BRB — > be right back*). We also identify informal intensifiers such as all-caps (e.g., *I LOVE this show!!!* and character repetitions (e.g., *I've got a mortgage!! happyyyyyy"*), note their presence in the tweet. All-caps words are made into lower case, and instances of repeated characters are replaced by a single character. Finally, the presence of any special Twitter tokens is noted (e.g., #hashtags, usertags, and URLs) and placeholders indicating the token type are substituted. Our hope is that this normalization improves the performance of the POS tagger, which is the last preprocessing step.

Part-of-speech:

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

Classification Algorithem:

Let's build a sentiment analysis of Twitter data to show how you might integrate an algorithm like this into your applications. We'll first start by choosing a topic, then we will gather tweets with that keyword and perform sentiment analyis on those tweets. We'll end up with an overall impression of whether people view the topic positively or not.

<u>Step 1:</u> 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);
});
```

first we use the Algorithmia API to pass our topic to the algorithm RetrieveTweetsWithKeyword(Retrieve tweets that include keyword anywhere int their text. Limited to 500 tweets per call.) as our input. This will grab tweets containing our phrase. Second, we clear out the retweets so that we don't have duplicate data throwing off our scores. Twitter conveniently includes "RT" at the beginning of each tweet, so we find tweets with that string and remove them from our data set. This leaves us with a convenient set of tweets in the array no_retweets.

Step 2: Perform Sentiment Analysis on Tweets
After gathering and cleaning our data set, we are ready to execute the sentiment analysis algorithm on each tweet. Then, we will calculate an average score for all the tweets combined.

```
var analyze_tweets = function(no_retweets) {
   var total_score = 0;
   var score_count = 0;
   var final_score = 0;

   // Execute sentiment analysis on every tweet in the array, then calculate average score.
   for (var j = 0; j < no_retweets.length; j++) {
      client.algo("nlp/SentimentAnalysis/0.1.1").pipe(no_retweets[j]).then(function(output) {
      if(output.error) {
         console.log(output.error);
      } else {
         console.log(output.result);
         score_count = score_count + 1;
         total_score = total_score + output.result;
      }
      // Calculate average score.
      if (score_count == no_retweets.length) {
         final_score = total_score / score_count;
         console.log('final score: ' + final_score);
      }
   })
  }
}
```

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.

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

For the following Sentiment Analysis with the help of the Survey we create a code to retrieve the data of tweeter, the code is :

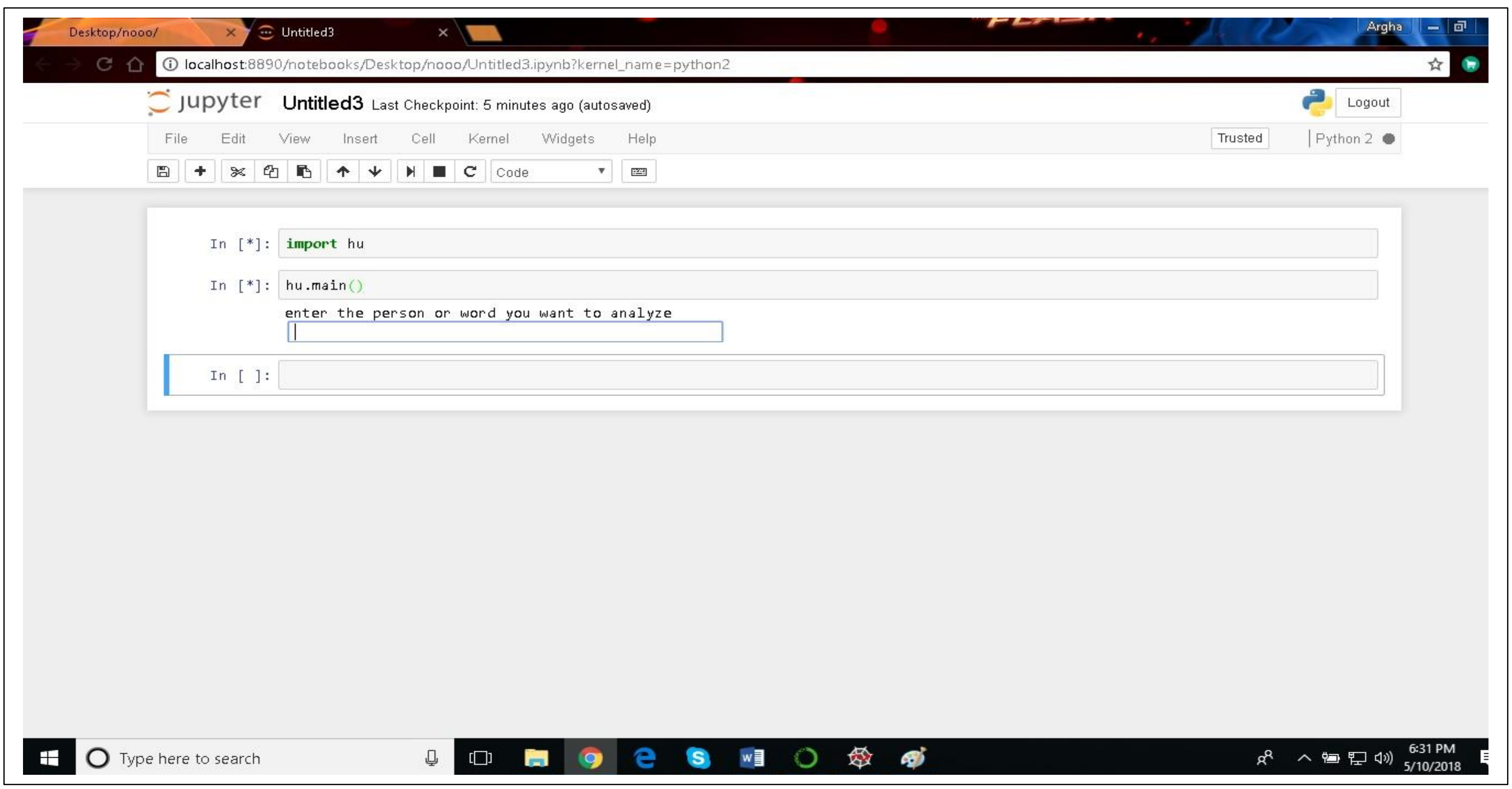
```
import re
import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob
import matplotlib.pyplot as plt
class TwitterClient(object):

    def __init__(self):


        consumer_key = 'jbkZ2RmduoH7eh9WBpPUQWro3'
        consumer_secret = 'RpdmaKXShnzz8NdjlBG6vad88E3CebNtx5Gb03fkEvGCst1fIo'
        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):


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

    def get_tweet_sentiment(self, tweet):


        analysis = TextBlob(self.clean_tweet(tweet))

        if analysis.sentiment.polarity > 0:
            return 'positive'
```

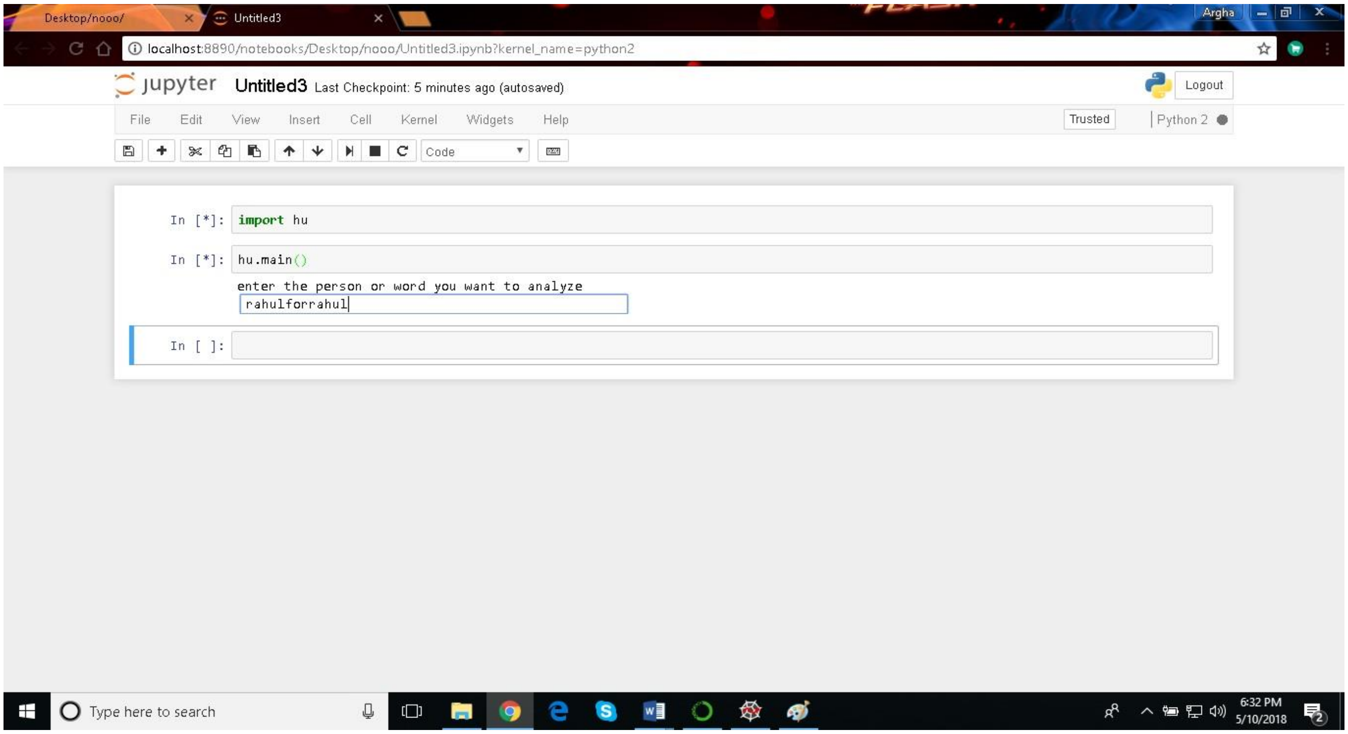
```
        elif analysis.sentiment.polarity == 0:
            return 'neutral'
        else:
            return 'negative'

    def get_tweets(self, query, count = 20000):

        tweets = []

        try:

            fetched_tweets = self.api.search(q = query, 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()

    tweets = api.get_tweets(query = raw_input("enter the person or word you want to analyze\n"),
count = 20000)
    ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']
    pt=format(100*len(ptweets)/len(tweets))
    print("POSITIVE TWEET PERSENTAGE: ",pt)
```

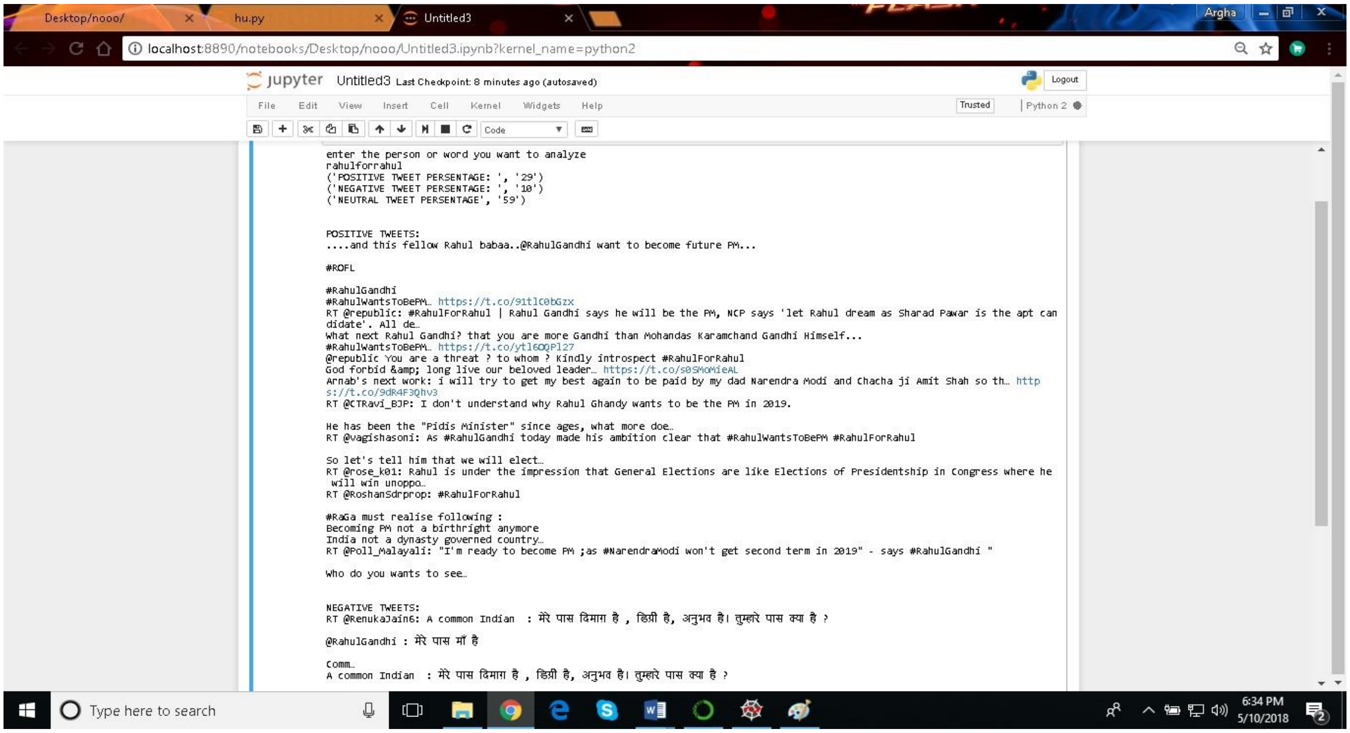
```
    ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']
    nt=format(100*len(ntweets)/len(tweets))
    print("NEGATIVE TWEET PERSENTAGE: ",nt)
    nut=format(100*(len(tweets) - len(ntweets) - len(ptweets))/len(tweets))
    print("NEUTRAL TWEET PERSENTAGE",nut)
    print("\n\nPOSITIVE TWEETS:")
    for tweet in ptweets[:10]:
        print(tweet['text'])
    print("\n\nNEGATIVE TWEETS:")
    for tweet in ntweets[:10]:
        print(tweet['text'])
    choice=raw_input("PIECHART\n")
    if choice=="" :
      labels = 'Positive', 'Negative', 'Neutral'
      sizes = [pt,nt,nut]
      colors = ['green', 'red', 'blue']
      explode = (0.1,0.1,0.1)  # explode 1st slice
      plt.pie(sizes, explode=explode, labels=labels, colors=colors,
      autopct='%1.1f%%', shadow=True, startangle=40)
      plt.axis('equal')
      plt.show()
def main():



    plot()


if __name__ == "__main__":

    main()
```

Results and Discussion:

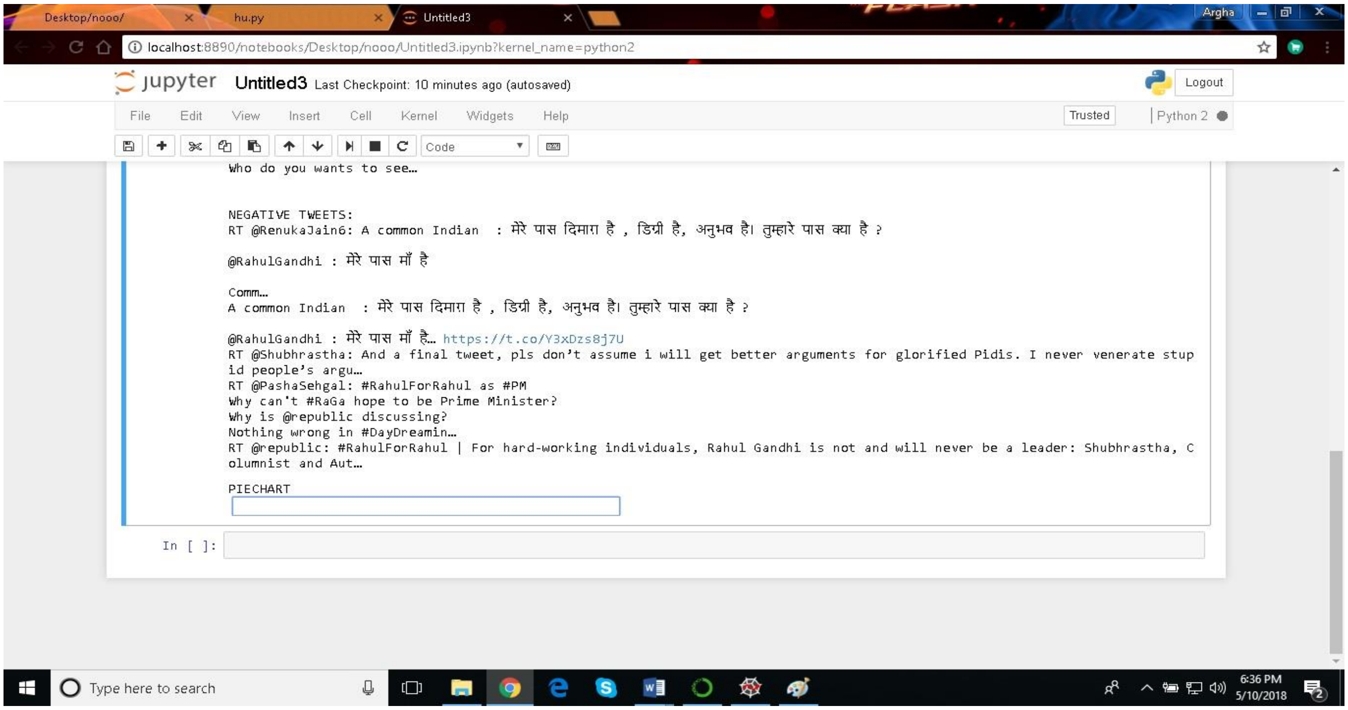
We will first present our results for the objective / subjective and positive / negative classifications. These results act as the first step of our classification approach. We only use the short-listed features for both of these results. This means that for the objective / subjective classification 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. Furthermore all the figures reported are the result of 10-fold cross validation. We take an average of each of the 10 values we get from the cross validation.
we make a condition while reporting the results of polarity classification (which differentiates between positive and negative classes) that only subjective labelled tweets are used to calculate these results. However, in case of final classification approach, any such condition is removed and basically both objectivity and polarity classifications are applied to all tweets regardless of whether they are labelled objective or subjective.



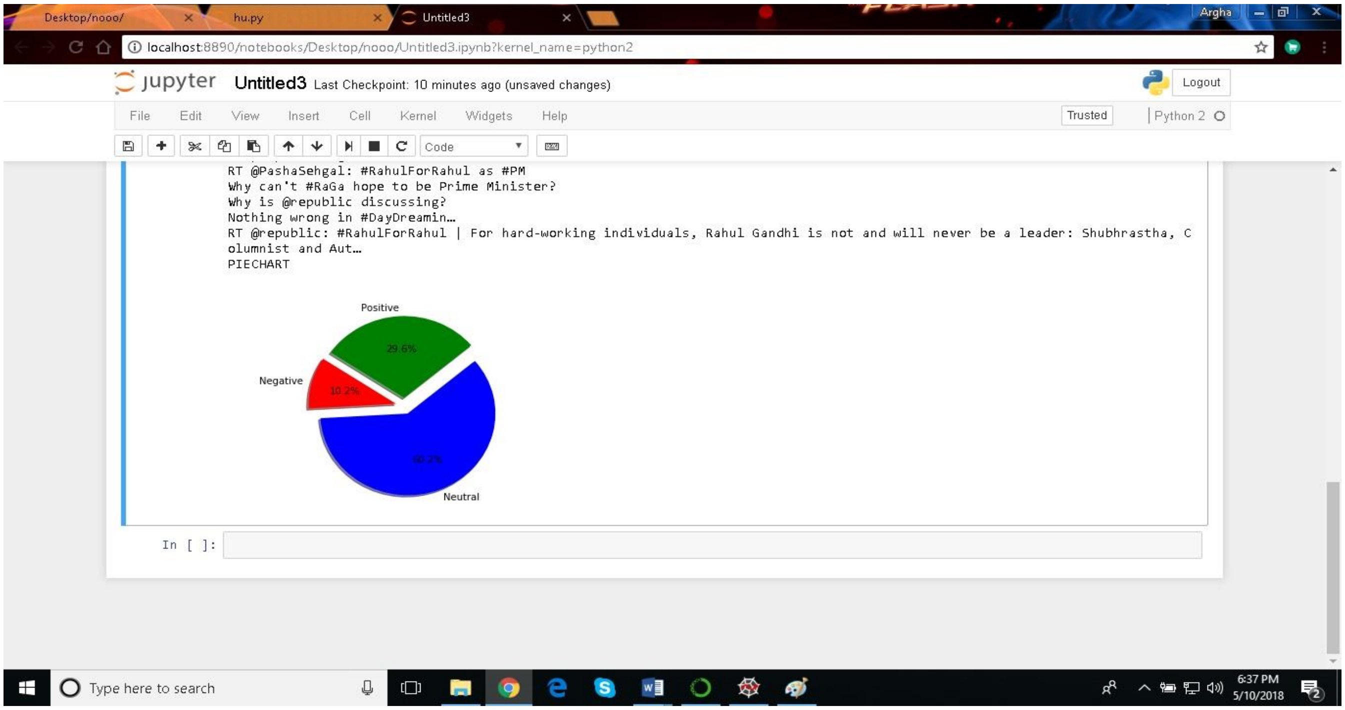

Desktop/nooo/
Untitled3
localhost:8890/notebooks/Desktop/nooo/Untitled3.ipynb?kernel_name=python2
jupyter Untitled3 Last Checkpoint: 5 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 2
Code
In [*]: import hu
In [*]: hu.main()
enter the person or word you want to analyze
rahulforrahul
In [ ]:
Type here to search
6:32 PM
5/10/2018



Desktop/nooo/
hu.py
Untitled3
Argha
localhost:8890/notebooks/Desktop/nooo/Untitled3.ipynb?kernel_name=python2
jupyter Untitled3 Last Checkpoint: 8 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 2
Code
enter the person or word you want to analyze
rahulforrahul
('POSITIVE TWEET PERSENTAGE: ', '29')
('NEGATIVE TWEET PERSENTAGE: ', '10')
('NEUTRAL TWEET PERSENTAGE', '59')
POSITIVE TWEETS:
....and this fellow Rahul babaa..@RahulGandhi want to become future PM...
#ROFL
#RahulGandhi
#RahulWantsToBePM… https://t.co/91tlC0bGzx
RT @republic: #RahulForRahul | Rahul Gandhi says he will be the PM, NCP says 'let Rahul dream as Sharad Pawar is the apt can
didate'. All de…
What next Rahul Gandhi? that you are more Gandhi than Mohandas Karamchand Gandhi Himself...
#RahulWantsToBePM… https://t.co/ytl6OQPl27
@republic You are a threat ? to whom ? Kindly introspect #RahulForRahul
God forbid & long live our beloved leader… https://t.co/s0SMoMieAL
Arnab's next work: i will try to get my best again to be paid by my dad Narendra Modi and Chacha ji Amit Shah so th… http
s://t.co/9dR4F3Qhv3
RT @CTRavi_BJP: I don't understand why Rahul Ghandy wants to be the PM in 2019.
He has been the "Pidis Minister" since ages, what more doe…
RT @vagishasoni: As #RahulGandhi today made his ambition clear that #RahulWantsToBePM #RahulForRahul
So let's tell him that we will elect…
RT @rose_k01: Rahul is under the impression that General Elections are like Elections of Presidentship in Congress where he
will win unoppo…
RT @RoshanSdrprop: #RahulForRahul
#RaGa must realise following :
Becoming PM not a birthright anymore
India not a dynasty governed country…
RT @Poll_Malayali: "I'm ready to become PM ;as #NarendraModi won't get second term in 2019" - says #RahulGandhi "
Who do you wants to see…
NEGATIVE TWEETS:
RT @RenukaJain6: A common Indian : मेरे पास दिमाग है , डिग्री है, अनुभव है। तुम्हारे पास क्या है ?
@RahulGandhi : मेरे पास माँ है
Comm…
A common Indian : मेरे पास दिमाग है , डिग्री है, अनुभव है। तुम्हारे पास क्या है ?
Type here to search
6:34 PM
5/10/2018



Desktop/nooo/
hu.py
Untitled3
Argha
localhost:8890/notebooks/Desktop/nooo/Untitled3.ipynb?kernel_name=python2
jupyter Untitled3 Last Checkpoint: 10 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 2
Code
Who do you wants to see…
NEGATIVE TWEETS:
RT @RenukaJain6: A common Indian : मेरे पास दिमाग है , डिग्री है, अनुभव है। तुम्हारे पास क्या है ?
@RahulGandhi : मेरे पास माँ है
Comm…
A common Indian : मेरे पास दिमाग है , डिग्री है, अनुभव है। तुम्हारे पास क्या है ?
@RahulGandhi : मेरे पास माँ है… https://t.co/Y3xDzs8j7U
RT @Shubhrastha: And a final tweet, pls don't assume i will get better arguments for glorified Pidis. I never venerate stup
id people's argu…
RT @PashaSehgal: #RahulForRahul as #PM
Why can't #RaGa hope to be Prime Minister?
Why is @republic discussing?
Nothing wrong in #DayDreamin…
RT @republic: #RahulForRahul | For hard-working individuals, Rahul Gandhi is not and will never be a leader: Shubhrastha, C
olumnist and Aut…
PIECHART
In [ ]:
Type here to search
6:36 PM
5/10/2018



If we compare these results to those provided by Wilson et al we see that although the accuracy of neutral class falls from 82.1% to 73% if we use our classification instead of theirs. However, for all other classes we report significantly greater results. Although the results presented by Wilson et al. are not from Twitter data they are of phrase level sentiment analysis which is very close in concept to Twitter sentiment analysis.

Finally we conclude that our classification approach provides improvement in accuracy by using even the simplest features and small amount of data set. However there are still a number of things we would like to consider as future work which we mention in the next section.

Conclusion and Future Scope:

The task of sentiment analysis, especially in the domain of micro-bloging, is still in the developing stage and far from complete. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance.

Right 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) under consideration and 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 should affect 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.
Apart from this, we are currently only focusing on unigrams and the effect of bigrams and trigrams may be explored. As reported in the literature review section when bigrams are used along with unigrams this usually enhances performance. However for bigrams and trigrams to be an effective feature we need a much more labeled data set than our meager 9,000 tweets.

Right now we are exploring Parts of Speech separate from the unigram models, we can try to incorporate POS information within our unigram models in future. So say instead of calculating a single probability for each word like P(word | obj) we could instead have multiple probabilities for each according to the Part of Speech the word belongs to. For example we may have P(word | obj, verb), P(word | obj, noun) and P(word | obj, adjective). Pang et al. used a somewhat similar approach and claims that appending POS information for every unigram results in no significant change in performance (with Naive Bayes performing slightly better and SVM having a slight decrease in performance), while there is a significant decrease in accuracy if only adjective unigrams are used as features. However these results are for classification of reviews and may be verified for sentiment analysis on micro blogging websites like Twitter.

One more feature we 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. Although Pang et al. 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 extremely simple model.

In this research we are focussing on general sentiment analysis. There is potential of work in the field of sentiment analysis with partially known context. For example we noticed that users generally use our website 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. So we can 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 if we apply general sentiment analysis on it instead.

Reference:

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

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

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

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

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

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

[7] 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