SENTIMENT ANALYSIS FOR SOCIAL NETWORKS

A MINI PROJECT REPORT

Submitted by

P BABURAJKUMAR

K U HARSHA VARDHAN

M JEYA SRINIVASAN

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PSNA COLLEGE OF ENGINEERING AND TECHNOLOGY, DINDIGUL

ANNA UNIVERSITY: CHENNAI 600025

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project report "**SENTIMENT ANALYSIS FOR SOCIAL NETWORKS**" is the bonafide work of **"P BABURAJKUMAR (921319205012), K U HARSHA VARDHAN (921319205040), M JEYA SRINIVASAN (921319205051)"** who carried out the project work under my supervision.

SIGNATURE OF HOD

Dr. A. Vincent Antony Kumar, M.E., Ph.D,

Professor And Head,

Department of IT,

PSNA College of Engineering and Technology, Dindigul.

SIGNATURE OF SUPERVISOR

Dr.K.Selvaraj M.E.,Ph.D.

Professor,

Department of IT,

PSNA College of Engineering and Technology, Dindigul.

Submitted for Viva-Voice examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to Lord Almighty, our supreme guide for bestowing his blessings upon our entire endeavour.

We take this opportunity to express our sincere thanks to the founder of our Institution(late) **Thiru.R.S.Kothandaraman**, and our beloved Chairperson **Tmt.K.Dhanalakshmiammal,** Pro-chairman **Rtn.MPHF R.S.K.Raguraam,** who are the guiding lights for all the activities in our college.

Our heartfelt gratitude and respect goes to our Principal, **Dr.D.Vasudevan M.E., Ph.D.,** for his wholehearted support and help in the completion of our project. We are extremely thankful to **Dr.A.Vincent Antony Kumar M.E., Ph.D.**, Head of the Department, Department of Information Technology, for providing all the necessary facilities for the successful completion of our project.

We extend our profound gratitude to our guide **Dr.K.Selvaraj M.E., Ph.D.** and project coordinator **Dr.R.Divya M.E., Ph.D.** Department of Information Technology under their guidance this project has attained every step of success.

We extend our heartfelt salutations to our beloved parents and faculty to establish this project in successful manner.

ABSTRACT

In today's world, Social Networking website like Twitter, Facebook, Tumbler, etc. plays a very significant role. Twitter is a micro-blogging platform which provides a tremendous amount of data which can be used for various applications of Sentiment Analysis like predictions, reviews, elections, marketing, etc. Sentiment Analysis is a process of extracting information from large amount of data, and classifies them into different classes called sentiments.

Python is simple yet powerful, high-level, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data by using Natural Language Toolkit (NLTK). NLTK is a library of python, which provides a base for building programs and classification of data. NLTK also provide graphical demonstration for representing various results or trends and it also provide sample data to train and test various classifiers respectively.

The goal of this thesis is to classify twitter data into sentiments (positive or negative) by using different supervised machine learning classifiers on data collected for different Indian political parties and to show which political party is performing best for public. We also concluded which classifier gives more accuracy during classification.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DOMAIN

In this chapter we are going to give the introductions on Sentiment Analysis, Python and Natural Language Toolkit (NLTK). Then we are explaining the objective of our thesis. After this we will discuss why there is a need of sentiment analysis and some of the applications of Sentiment Analysis which are used in our daily life.

Introduction to Sentiment Analysis:

Sentiment Analysis is process of collecting and analyzing data based upon the person feelings, reviews and thoughts. Sentimental analysis often called as opinion mining as it mines the important feature from people opinions. Sentimental Analysis is done by using various machine learning techniques, statistical models and Natural Language Processing (NLP) for the extraction of feature from a large data.

Sentiment Analysis can be done at document, phrase and sentence level. In document level, summary of the entire document is taken first and then it is analyze whether the sentiment is positive, negative or neutral. In phrase level, analysis of phrases in a sentence is taken in account to check the polarity. In Sentence level, each sentence is classified in a particular class to provide the sentiment.

Sentimental Analysis has various applications. It is used to generate opinions for people of social media by analyzing their feelings or thoughts which they provide in form of text. Sentiment Analysis is domain centered, i.e. results of one domain cannot be applied to other domain. Sentimental Analysis is used in many real life scenarios, to get reviews about any product or movies, to get the financial report of any company, for predictions or marketing.

Twitter is a micro blogging platform where anyone can read or write short form of message which is called tweets. The amount of data accumulated on twitter is very huge. This data is unstructured and written in natural language. Twitter Sentimental Analysis is the process of accessing tweets for a particular topic and predicts the sentiment of these tweets as positive, negative or neutral with the help of different machine learning algorithm.

1.2 Introduction to Python

Python is a high level, dynamic programming language which is used for this thesis. Python3.4 version was used as it is a mature, versatile and robust programming language. It is an interpreted language which makes the testing and debugging extremely quickly as there is no compilation step. There are extensive open source libraries available for this version of python and a large community of users.

Python is simple yet powerful, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data, i.e. spoken English using NLTK. Other high level programming languages such as 'R' and 'Matlab' were considered because they have many benefits such as ease of use but they do not offer the same flexibility and freedom that Python can deliver.

1.3 Introduction to NLTK

Natural Language Toolkit (NLTK) is library in Python, which provides a base for building programs and classification of data. NLTK is a collection of resources for Python that can be used for text processing, classification, tagging and tokenization. This toolbox plays a key role in transforming the text data in the tweets into a format that can be used to extract sentiment from them.

NLTK provides various functions which are used in pre-processing of data so that data available from twitter become fit for mining and extracting features. NLTK support various machine learning algorithms which are used for training classifier and to calculate the accuracy of different classifier.

In our thesis we use Python as our base programming language which is used for writing code snippets. NLTK is a library of Python which plays a very important role in converting natural language text to a sentiment either positive or negative. NLTK also provides different sets of data which are used for training classifiers. These datasets are structured and stored in library of NLTK, which can be accessed easily with the help of Python.

1.4 Introduction to Supervised Machine learning Classifiers

Supervised machine learning is a technique whose task is to deduce a function from tagged training samples. The training samples for supervised learning consist of large set of examples for a particular topic. In supervised learning, every example training data comes in a pair of input (vector quantity) and output value (desired result). These algorithms analyze data and generate an output function, which is used to mapped new data sets to respective classes. Different machine learning classifiers which we are going to use to build our classifier are:

∑ Naïve-Bayes Classifier.

∑ MultinomialNB Classifier.

∑ BernoulliNB Classifier.

∑ Logistic Regression Classifier.

∑ SGDC (Stochastic Gradient Decent Classifier).

∑ SVC (Support Vector Classifier): LinearSVC and NuSVC.

CHAPTER 2

LITERATURE SURVEY

Many research have been done on the subject of sentiment analysis in past. Latest research in this area is to perform sentiment analysis on data generated by user from many social networking websites like Facebook, Twitter, Amazon, etc. Mostly research on sentiment analysis depend on machine learning algorithms, whose main focus is to find whether given text is in favor or against and to identify polarity of text. In this chapter we will provide insight of some of the research work which helps us to understand the topic deep.

P. Pang, L. Lee, S. Vaithyanathan et al [8]

They were the first to work on sentiment analysis. Their main aim was to classify text by overall sentiment, not just by topic e.g., classifying movie review either positive or negative. They apply machine learning algorithm on movie review database which results that these algorithms out-perform human produced algorithms. The machine learning algorithms they use are Naïve-Bayes, maximum entropy, and support vector machines. They also conclude by examining various factors that classification of sentiment is very challenging. They show supervised machine learning algorithms are the base for sentiment analysis.

P. Pang, L. Lee et al [9]

By collecting large amount of data has always been a key to find out what people is thinking or expecting. With the emergence in the field of social media, availability of data which is full of opinion resources is very high. Other resources such as blogs,review sites, messages, etc. are helping us to know what people can do and their opinion about the topic. The sudden increase of work in the field of data mining and sentiment extraction deals with the computational power to solve the problem of opinion mining or subjectivity in text. Hence various new systems are created based on different languages and commands that can deal directly with opinion mining as the first class object and direct response or live research also becoming the area of interest.

They take a survey which covers that methodology and approaches that are used in direct response of opinion mining are more helpful than others. Their focus is on functions that can solve new challenges rising in sentiment analysis applications. They also compared these new techniques to already present traditional analysis which is based on facts.

E. Loper, S. Bird et al [10]

Natural Language Toolkit (NLTK) is a library which consists of many program modules, large set of structured files, various tutorials, problem sets, many statistics functions, ready-to-use machine learning classifiers, computational linguistics courseware, etc. The main purpose of NLTK is to carry out natural language processing, i.e. to perform analysis on human language data. NLTK provides corpora which are used for training classifiers. Developers create new components and replace them with existing component, more structured programs are created and more sophisticated results are given by dataset.

H. Wang, D. Can, F. Bar, S. Narayana et al [11]

They were the researchers who proposed a system for real time analysis of public responses for 2012 presidential elections in U.S. They collect the responses from Twitter, a micro blogging platform. Twitter is one the social network site where people share their views, thoughts and opinions on any trending topic. People responses on Twitter for election candidates in U.S. created a large amount of data, which helps to create a sentiment for each candidate and also created a prediction of whom winning.

A relation is created between sentiments that arise from people response on twitter with the complete election events. They also explore how sentiment analysis affects these public events. They also show this live sentiment analysis is very fast as compared to traditional content analysis which takes many days or up to some weeks to complete. The system they demonstrated analyzes sentiment of entire Twitter data about the election, candidates, promotions, etc. and delivering results at a continuous.

L. Jiang, M. Yu, M. Zhou, X. Liu, T. Zhao et al [13]

Twitter sentiment analysis was growing at faster rate as amount of data is increasing. They created a system which focuses on target dependent classification. It is based on Twitter in which a query is given first; they classify the tweets as positive, negative or neutral sentiments with respect to that query that contain sentiment as positive,

Also, when state-of-the-art approaches are used for classification they only take tweet into consideration. These approaches ignore related tweet, as they classify based on current tweet.

However, because tweets have property to be short and mostly ambiguous, considering current tweet only for sentiment analysis is not enough. They propose a system to improve target-dependent Twitter sentiment classification by:

1) Integrating target-dependent features, and

2) Taking related tweets into consideration.

According to their experimental results, these new advancement highly improves the efficiency and performance of target-dependent sentiment classification.

C. Tan, L. Lee, J. Tang, L. Jiang, M. Zhou, P. Li, et al [14]

They show that information that can be used to improve user-level sentiment analysis. There base of research is social relationships, i.e. users that are connected in any social platform will somehow hold similar opinions, thoughts; therefore, relationship information can supplement what they extract from user's viewpoint. They use Twitter as there source of experimental data and they use semi-supervised machine learning framework to carry out analysis. They propose systems that are persuade either from the network of Twitter followers or from the network formed by users in Twitter in which users referring to each other using "@username". According to them, these semi-supervised learning results shows that by including this social network information leads to statistically significant improvement in performance of sentiment analysis classification over the performance based on the approach of SVM (Support Vector Machines) that have only access to textual features.

A. Pak, P. Paroubek et al [15]

Micro-blogging nowadays has become very popular communication platform among users in social network. Billions of tweets share every year among millions of users, in which they share opinions, feelings on different aspects of daily life. Thus micro blogging websites like Twitter, Friendfeed, Tumbler, etc. are rich sources of data for feature extraction and sentiment analysis. They also use Twitter one of the most popular micro-blogging website, for the implementation of sentiment analysis. They automatically collect a corpus (database) for training classifier to carry out sentiment analysis and opinion mining.

They perform linguistic inspection of collected corpus and build a sentiment classifier that is used to determine positive, negative and neutral sentiments of twitter document. They also proposed a system for emoticons used in tweets, in which they create a corpus for emoticons such that they can replace each emoticon with their respective meaning so that it can extract feature from emoticons. An example of emoticons is shown is Table 2.1, experimental calculations show that their proposed techniques are more efficient and give more performance than previous proposed models.

B. Sun, V. Ng, et al [16]

Many efforts have been done to gather information from social networks to perform sentiment analysis on internet users. Their aim is to show how sentimental analysis influences from social network posts and they also compare the result on various topics on different social-media platforms. Large amount of data is generated every day, people are also very curious in finding other similar people among them. Many researchers' measures the influence of any post through the number of likes and replies it received but they are not sure whether the influence is positive or negative on other post. In their research some questions are raised and new methodologies are prepared for sentimental influence of post.

CHAPTER 3

SOFTWARE REQUIREMENTS

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.

3.1 External Interface Requirement:

The project 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.

3.1.1 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.

3.1.2 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.

3.1.3 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.

3.2 Non Functional Requirement:

3.2.1 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.

3.2.2 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.

3.2.3 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.

3.2.4 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.

3.2.5Other Requirements:

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

3.3 DESIGN:

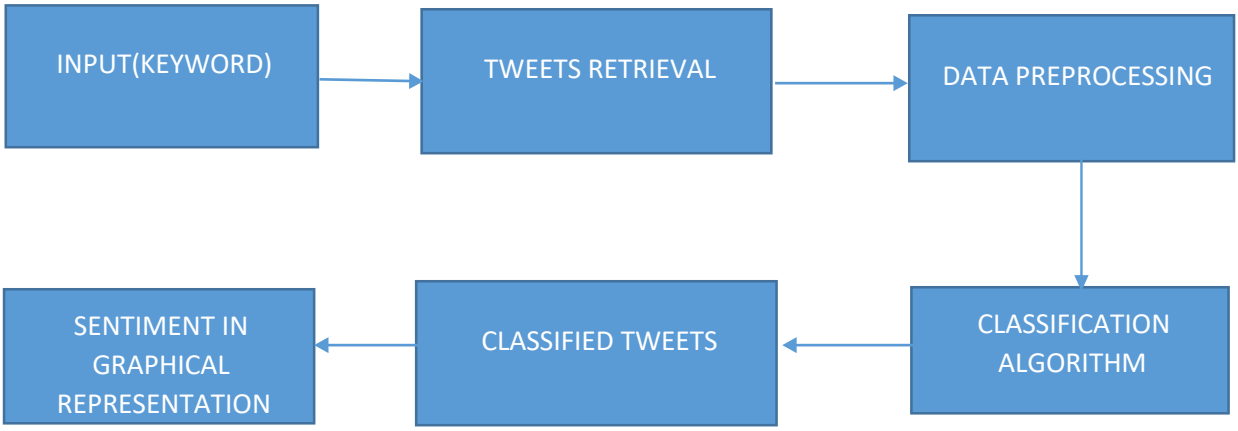


Fig.3.1 Shows overview of sentimental analysis

CHAPTER 4

PROBABILITY ALGORITHMS

4.1 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, Words like great, richly, awesome, and pathetic, and awful and ridiculously are very informative cues:

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

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

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

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

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

$c = \text{argmax } P(c|d)$ 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't matter)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:

Table 4.1. Polarity , Subjectivity , Intensity for the word “Great”:

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:

Table 4.2. Polarity , Subjectivity , Intensity for the word “Very”:

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.

CHAPTER 5

IMPLEMENTATION AND RESULT

5.1 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".

5.1.1 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.

5.1.2 Data Pre processing:

Data pre processing 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, user tags, 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 pre processing 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.

5.1.3 Classification Algorithm:

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

Step 1: Gather Tweets

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

```
var algorithmia = require("algorithmia");
var client = algorithmia(process.env.ALGORITHMIA_API_KEY);
var input = "happy";
var no_retweets = [];

console.log("Analyzing tweets with phrase: " + input);
client.algo("/diego/RetrieveTweetsWithKeyword/0.1.2").pipe(input).then(function(output) {
if (output.error) {
      console.log(output.error);
   } else {
var tweets = [];
var tweets = output.result;
for (var i = 0; i < output.result.length; i++) {
        // Remove retweets. All retweets contain "RT" in the string.
if (tweets[i].indexOf('RT') == -1) {
          no_retweets.push(tweets[i]);
        }
   }
}
// We will cover this function in the next step.
analyze_tweets(no_retweets);
});
```

First algorithamia API passes 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()
```

CHAPTER 6

CONCLUSION

This project proposes the results for the objective / subjective and positive / negative classifications. These results act as the first step of our classification approach. This project only use the short-listed features for both of these results. This means that for the objective / subjective classification it has 5 features and for positive / negative classification it has 3 features.

For both of these results this project use the Naïve Bayes classification algorithm, because that is the algorithm of this project are employing in our actual classification approach at the first step. Furthermore all the figures reported are the result of 10-fold cross validation. The goal of this project is to take an average of each of the 10 values and get from the cross validation.

This project makes 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.

CHAPTER 7

FUTURE WORK

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

Right now the project is working with only the very simplest unigram models; The project can improve those models by adding extra information like closeness of the word with a negation word. The project 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, the project currently focuses 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 the project needs much more labeled data set than our meager 9,000 tweets.

Right now the project are exploring Parts of Speech separate from the unigram models, it 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) this could instead have multiple probabilities for each according to the Part of Speech the word belongs to. For example this project 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 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 project it focussing on general sentiment analysis. There is potential of work in the field of sentiment analysis with partially known context. For example it is 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 the project 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 with the general sentiment analysis on it instead.

REFERENCES

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.

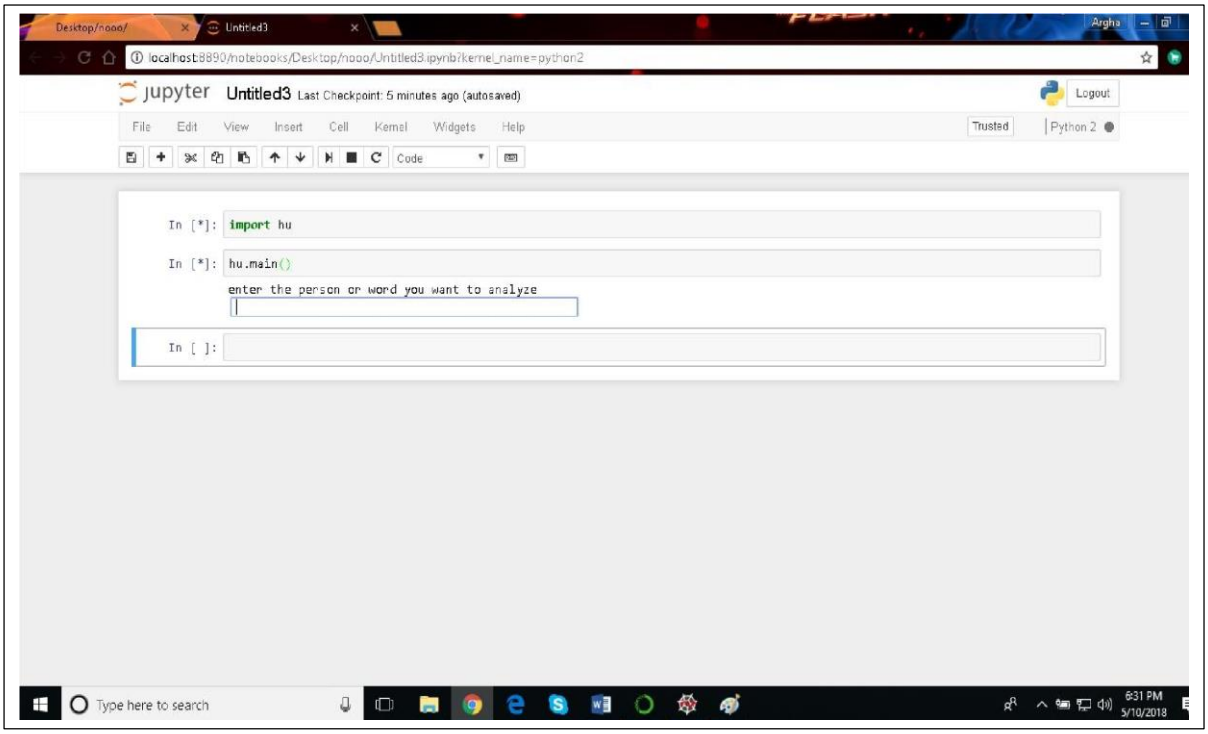
5. Manju Venugopalan and Deepa Gupta ,Exploring Sentiment Analysis on Twitter Data, IEEE 2015.

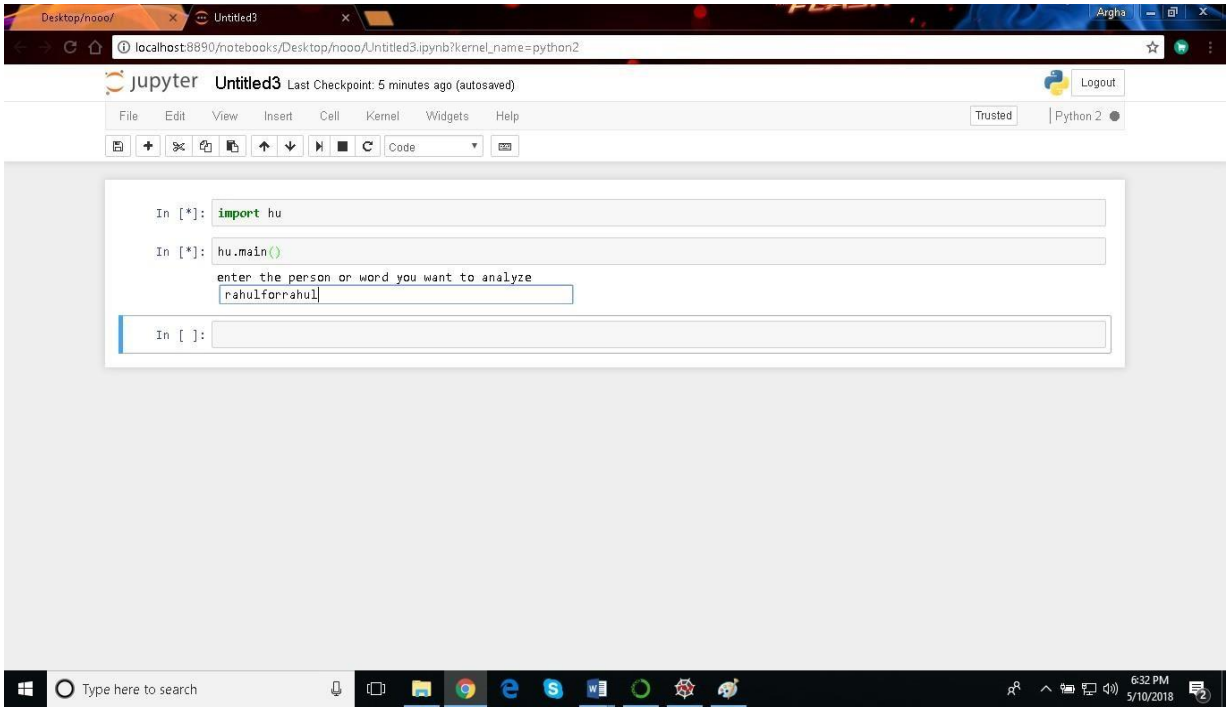
6. Brett Duncan and Yanqing Zhang, Neural Networks for Sentiment Analysis on Twitter.2017.

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.

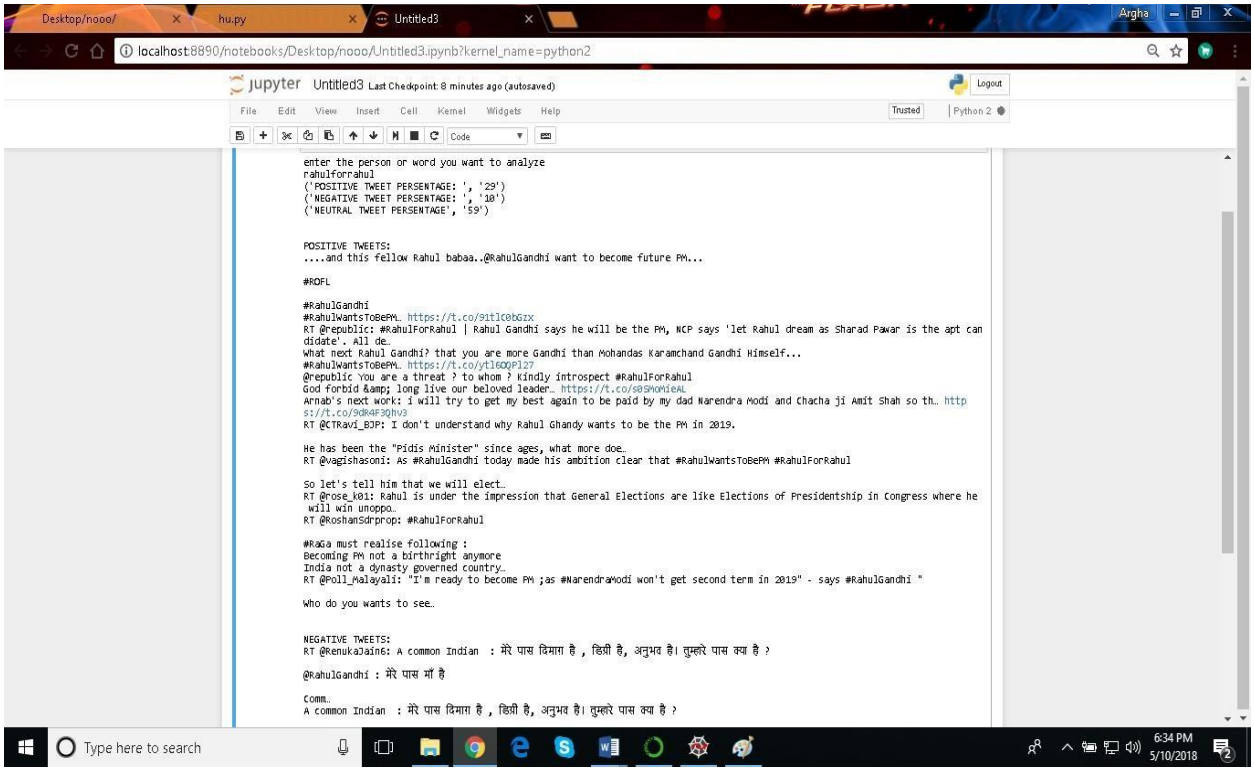
Sample Output:

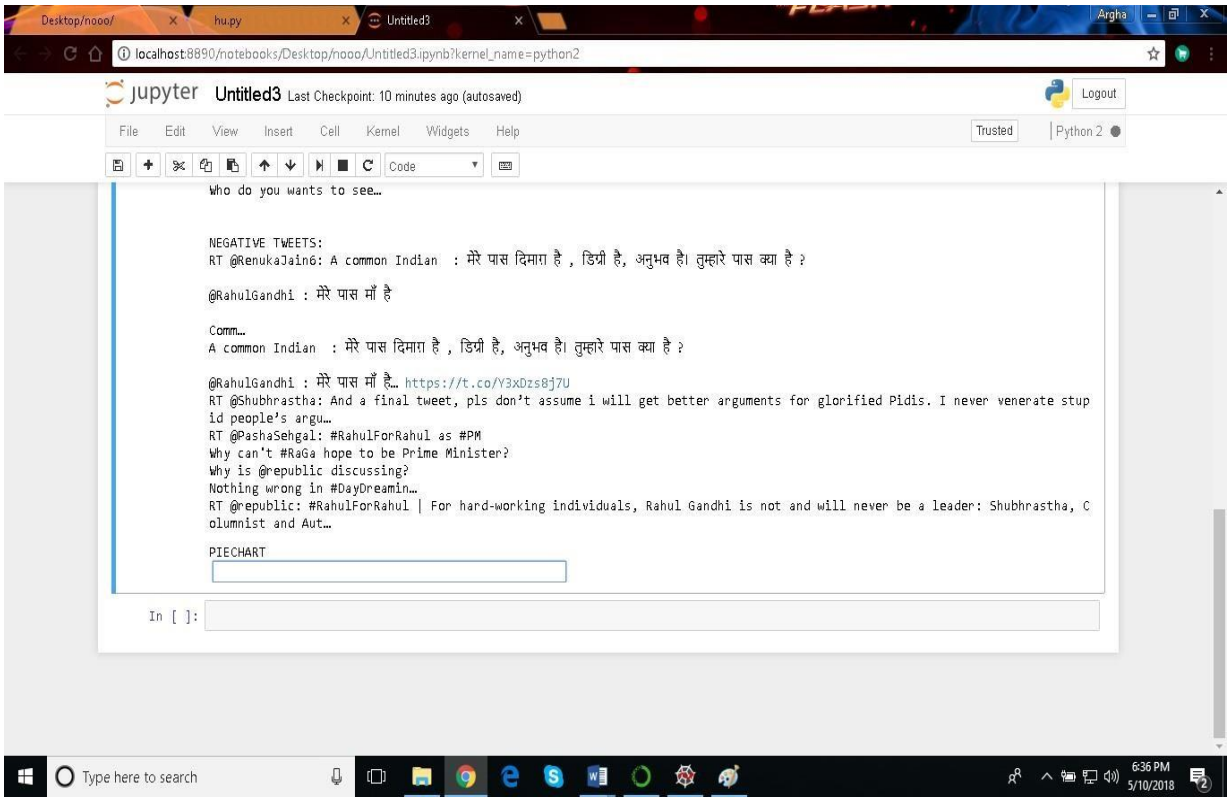
1)Entering the input:



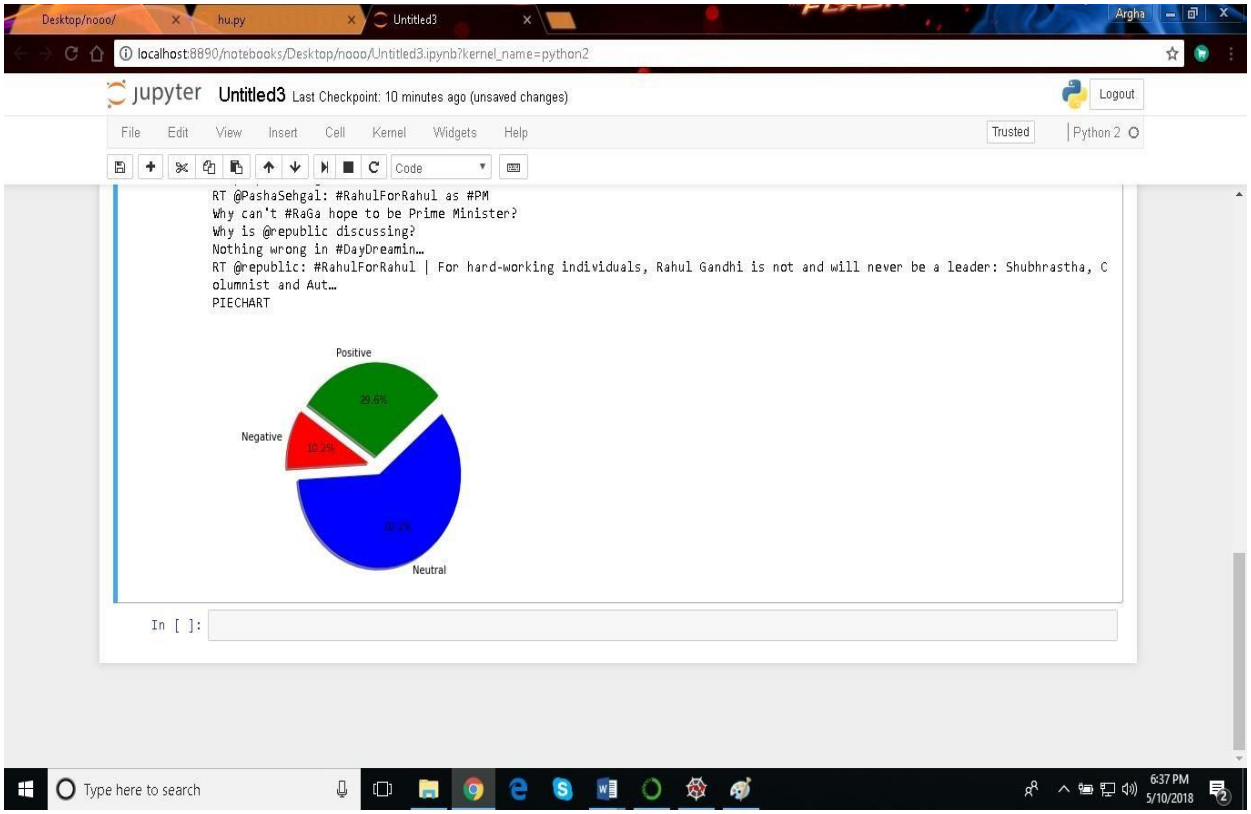


2)Classified tweets:





3)Visualisation of tweets:



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.