

**Emotion Classification in Social Networking Sites Using
Lexicon Analysis**

Project Report submitted in partial fulfillment of the requirement
for the degree of

Bachelor of Technology.

in

Information Technology

under the Supervision of

Dr. Pardeep Kumar

By

Samyak Handa, Roll No. 111411



Jaypee University of Information Technology
Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled “**Emotion Classification in Social Networking Sites using Lexicon Analysis**”, submitted by **Samyak Handa (Roll No. 111411)** in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date : 08.05.2015

Dr. Pardeep Kumar

**Assistant Professor
(Senior Grade)**

Acknowledgement

I express my sincere gratitude to my respected project supervisor Dr. Pardeep Kumar, Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat under whose supervision and guidance this work has been carried out. His whole hearted involvement, advice, support and constant encouragement throughout, have been responsible for carrying out this project work with confidence. I am also grateful to him for providing me with required infrastructural facilities that have been highly beneficial to me in undertaking the above mentioned project.

I am sincerely grateful to Brig. S.P. Ghrera, Professor and Head of Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat for providing all necessary facilities for the successful completion of my project.

Last but not the least, I would also like to thank my parents, Dr. (Mrs.) Garima Handa and Dr. Vivek Handa, and friends, Ms. Shivai Gupta, Mr. Anirudh Kaushik, Mr. Shubham Gupta, and Mr. Ujjwal Syal, for all their moral support and technical help, in bits and pieces, throughout the tenure of this project.

Date:08.05.2015

Samyak Handa

Table of Contents

| S.No. | Topic | Page No. |
|--------------|---|-----------------|
| 1 | Introduction | 8 |
| | 1.1 Problem Statement | 12 |
| | 1.2 Motivation | 12 |
| 2 | Literature Review | 13 |
| | 2.1 Sentiment Analysis | 14 |
| | 2.2 Classification Levels of sentiment analysis | 20 |
| | 2.3 Serendio: Simple and Practical lexicon based approach to Sentiment Analysis | 21 |
| | 2.4 Method Proposed | 25 |
| | 2.4.1 Extracting posts from Twitter | 25 |
| | 2.4.2 Using a Dictionary Based Approach (In Progress) | 30 |
| | 2.4.3 Sentiment analysis algorithms and applications: A survey | 30 |
| 3 | Methodology | 32 |
| 4 | Codes and Screenshots | 41 |
| | 4.1 Extracting Data from Twitter | 41 |
| | 4.2 Word Tokenization using NumPy and NLTK | 44 |
| | 4.3 Building the Word Base | 47 |
| | 4.4 Testing the word base on a sample document | 51 |
| | 4.5 Screenshots | 54 |
| 5 | Conclusion | 58 |
| | 5.1 Conclusion | 58 |
| | References | 59 |

List of Figures

| S.No. | Title | Page No. |
|--------------|---|-----------------|
| 1. | General Flow of Process of Sentiment Analysis | 11 |
| 2. | Flow Chart of Procedure | 19 |
| 3. | Algorithm used for preprocessing | 24 |
| 4. | Downloading Python | 32 |
| 5. | Getting PIP | 33 |
| 6. | Locating pip in PythonXX\Scripts | 33 |
| 7. | Downloading Tweepy | 34 |
| 8. | Creating an App | 34 |
| 9. | Registering and creating a Twitter App | 35 |
| 10. | Twitter App Drawer | 36 |
| 11. | Generating Access Token and Access Token Secret | 37 |
| 12. | The OAuth Tool | 38 |
| 13. | Text tokenization through Python and NLTK | 54 |
| 14. | Creating the chunked tree presenting POS and named entity | 55 |
| 15. | Adding words to the Word Base | 56 |
| 16. | Analyzing a document for sentiment | 57 |

List of Tables

| S.No. | Title | Page No. |
|--------------|------------------------------|-----------------|
| 1 | Social Media users worldwide | 10 |
| 2 | Python Introduction | 27 |

Abstract

Social Media is used by practically everybody. It is used as a medium to express freely, the thoughts, opinions, beliefs, behaviours and a lot more. This information in regard of a product, service or anything being used by people can serve a great purpose. It serves the purpose of a feedback mechanism. In terms of business, such feedback may be used to improve the service being provided, upgrade the product already in the market so that better releases can be used. In terms of education, such feedback information can be used for feedback of a course, or a teacher. Hence this project has been thought of. Right now, what has already been developed is the part where we extract data from a social networking site, Twitter, and the part where we try to make something meaningful out of the sentences that have been extracted by tokenizing them and identifying parts of speech. Also discussed in the report are the procedures to go about the above two steps and the future work of the project, which includes comparing this lexicon based approach with a machine learning based approach and then if possible, integrating this project with the Student Feedback System of Jaypee University of Information Technology, Wanknaghat.

CHAPTER 1 : INTRODUCTION

From the point of view of data mining, a **social network** is a *heterogeneous* and *multirelational* data set represented by a graph. The graph is typically very large, with **nodes** corresponding to *objects* and **edges** corresponding to *links* representing relationships or interactions between objects. Both nodes and links have *attributes*. Objects may have class labels. Links can be one-directional and are not required to be binary.

Social networks need not be social in context. There are many real-world instances of technological, business, economic, and biologic social networks. Examples include electrical power grids, telephone call graphs, the spread of computer viruses, the World Wide Web, and coauthorship and citation networks of scientists. Customer networks and collaborative filtering problems (where product recommendations are made based on the preferences of other customers) are other examples. In biology, examples range from epidemiological networks, cellular and metabolic networks, and food webs, to the neural network of the nematode worm *Caenorhabditis elegans* (the only creature whose neural network has been completely mapped). The exchange of e-mail messages within corporations, newsgroups, chat rooms, friendships, sex webs (linking sexual partners), and the quintessential “old-boy” network (i.e., the overlapping boards of directors of the largest companies in the United States) are examples from sociology.

Small world (social) networks have received considerable attention as of late. They reflect the concept of “small worlds,” which originally focused on networks among individuals. The phrase captures the initial surprise between two strangers (“What a small world!”) when they realize that they are indirectly linked to one another through mutual acquaintances. In 1967, social psychologist Stanley Milgram performed an experiment to solve an unresolved hypothesis circulating in those days. The hypothesis was called the small-world problem. The claim of the small-world phenomenon is that the world, in a sense small, when viewed as a network of social acquaintances, could be reached through a network of friends in a only a few steps. Milgram asked a few hundred randomly selected people to send letters to a stock broker in Boston via intermediaries. They can send the letter to people they knew on first name basis. Among the letters that reached the

destination correctly, the average path length was found to be six. This led to the phrase “six degrees of separation”. This experiment laid the stage for algorithmic aspects this new and emerging science.

In order to make such a claim, instead of asking, “How small is our world”, one could ask, “What would it take for any world to be small?” In other words, we want to construct a mathematical model of the world in which the individuals are represented as nodes and relationships are represented as edges. This allows analysis using tools of mathematics.

Imagine one has one hundred friends, each one of them also has hundred friends. So at one degree of separation one connects to one hundred people and at two degrees connects to one hundred times one hundred. Proceeding in a similar fashion, in five degrees he is connected to nine billion people. So if everyone has one hundred friends, then within six steps he can connect himself to the entire population.

In today’s era, almost four out of five users of internet use social media for some or other context. Some of these include friendship networks, blogging and micro-blogging sites, content and video sharing sites, e-commerce sites etc. The involvement and contribution of the users on the web is increasing day by day. One such contribution is reviews of users in social networking sites. The current trend of giving online reviews enables users to take better decisions who want to use a particular service or purchase a particular product. It helps them to check the popularity of the product. It also enables them to extract the positive or negative features of the products by reading reviews. But manual analysis of such a huge amount of reviews can lead to biased decision. So to provide automation, we are studying sentiment analysis. Sentiment analysis is the modern methodology which analyze huge amount of data to extract sentiments associated with the data. The growth of internet has a special significance in online service. Today, a large amount of population uses social media to give their reviews. The social media Universe is expanding.

| Platform | Monthly active Users |
|--------------------|-----------------------------|
| Facebook | 1.28 Billion |
| Twitter | 255 Million |
| Linkedin | 1.84 Million |
| Youtube | 1 Billion+ |
| Google Plus | 540 Million |

Table 1 : Social Media users worldwide

The use of social media is increasing day by day and this is represented by the number of monthly users in Table 1. Increasing growth of social media users over internet has also increased their participation in various discussions and activities simultaneously. In case of a product, reviews of users will help to take many important decisions about the services of the product. But manually reading such a bulk amount reviews is a very difficult task. So there is a need of an automatic system which will lead to automatically extract the positive and negative features of the product and make the decision making process easier. There are many sites which do this.

Sentiment analysis is a text classification problem which deals with extracting information present within the text. This extracted information can be then further classified according to its polarity as positive, negative or neutral. It can be defined as a computational task of extracting sentiments from the opinion. Some opinions represent sentiments and some opinions do not represent any sentiment.

- ▶ **Sentiments:** Opinions or in other sense can be recognized as someone's linguistic expressions of emotions, beliefs, evaluations etc.
- ▶ **Analysis:** To capture the opinions from a pool of users whether the opinion is positive, negative or neutral.
- ▶ **Benefit:** Provide efficient information in decision making

Example:

User's Opinion: Person a: it's a great movie (positive statement)

Person b: the new iphone is awesome..!!! (Positive statement)

Person c: Nah!! I didn't like it at all.. (Negative statement)

Polarity :

- Positive
- Negative
- Neutral

Reading huge amount of reviews and discussions over internet is not an easy task and finally to take decision. But these discussions and reviews help in many sectors such as improving e-learning environment, providing personalization in e-learning environment, for getting public response to governmental activities.

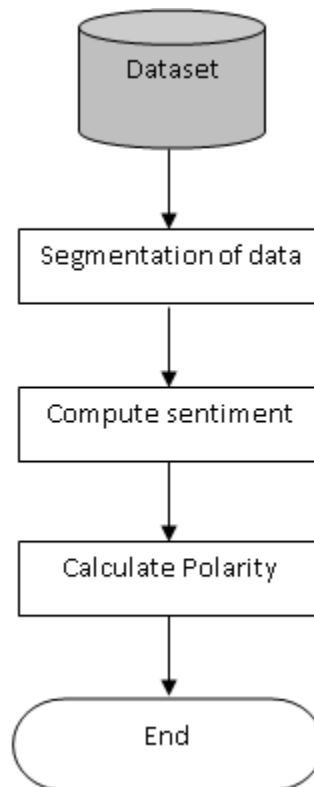


Figure 1 : General Flow of Process of Sentiment Analysis

1.1 Problem Statement

From the given dataset, what we have to do is to extract the data and segment that data according to parts of speech. After that we will check the sentiments and assign tags to the extracted tokens. In the last step overall polarity of the text is calculated. If the polarity of data is positive, it is positive sentence and if polarity is negative it is negative sentence.

1.2 Motivation

In today's era, almost four out of five users of internet use social media for some or other context. Some of these include friendship networks, blogging and micro-blogging sites, content and video sharing sites, e-commerce sites etc. The involvement and contribution of the users on the web is increasing day by day. One such contribution is reviews of users in social networking sites. In case of a product, reviews of users will help to take many important decisions about the services of the product. But manually reading such a bulk amount reviews is a very difficult task. So there is a need of an automatic system which will lead to automatically extract the positive and negative features of the product and make the decision making process easier. There are many sites which do this. But in case of education we can also use this sentiment analysis technique. In education, use of computers and internet leads to the explosion of study material for both teachers and students. This contribution of computers goes to e-learning systems and incorporation of adaptive methods and techniques goes to the development of adaptive e-learning systems. These adaptive e-learning systems will help each and every student in their personalized learning process. For providing personalization, information of each student should be stored in their personal student model. Sentiment analysis technique will help to extract student's emotional state for a particular subject and accordingly will make motivational steps to create the interest of the student. Different types of tasks can be assigned to the students according the polarity of their reviews about a particular subject in adaptive e-learning system. This system will help to know the potential needs of a student at a particular time. Also this adaptive e-learning system can act as a feedback for the teachers. According to the feedback, teachers can make the changes in methods and techniques used in learning process.

CHAPTER 2 : LITERATURE REVIEW

The use of computers in education has meant a great contribution for students and teachers. The incorporation of adaptation methods and techniques allows the development of adaptive elearning systems, where each student receives personalized guidance during the learning process (Brusilovsky, 2001). In order to provide personalization, it is necessary to store information about each student in what is called the student model (Kobsa, 2007). The specific information to be collected and stored depends on the goals of the adaptive e-learning system (e.g., preferences, learning styles, personality, emotional state, context, previous actions, and so on).

Knowing the users' emotions is useful not only in the educational context but also in many others (e.g., marketing, politics, online shopping, and so on) (Feldman, 2013). In general, in order for a system to be able to take decisions based on information about the users, it is necessary for it to get and store information about them. One of the most traditional procedures to obtain information about users consists of asking them to fill in questionnaires. However, the users can find this task too time-consuming. Recently, non-intrusive techniques are preferred (de Montjoye, Quoidbach, Robic, & Pentland, 2013). We also think that information for student models should be obtained as unobtrusively as possible, yet without compromising the reliability of the model built (Ortigosa, Carro, & Quiroga, 2013)

When reflecting about potential sources of information regarding user sentiment, we looked for digital places in which the users express themselves frequently and naturally. Nowadays, the number of users interacting with others through social networks is growing exponentially. Therefore, we focused on social networks. There exist an increasing number of online social networks available through the Web. From these applications, the social network chosen for this project is Twitter. It focuses on professional relationships, and serves as a source of information. Users are related to each other by the concept of *following*. A user can see the posts, called *tweets* of the users he is *following* and can make his *tweets* visible to the users who follow him/her. Users can also exchange personal text messages.

When dealing with users and sentiments, it is useful to know the users' emotional state at a certain time (positive/neutral/negative), in order to provide each of them with personalized assistance accordingly. Moreover, it is also interesting to know whether this state corresponds to their "usual state" or, on the contrary, a noticeable variation might have taken place. Behavior variations, as detected in the messages written by a user (when sentiment histories are available), can indicate changes in the user's mood, and specific actions could be potentially needed or recommended in such cases.

2.1 Sentiment Analysis

Sentiment analysis has been defined as the computational study of opinions, sentiments and emotions expressed in texts (Liu, 2010). For the sake of simplifying the development of an emotion recognition tool, we have tried to avoid complex and potentially controversial definitions of emotions and sentiments. In this direction, we take the simplified definition of sentiment as "a personal positive or negative feeling or opinion". An example of a sentence transmitting a positive sentiment would be "I love it!", whereas "It is a terrible movie" transmits a negative one. A neutral sentiment does not express any feeling (e.g. "I am commuting to work"). Most of works in this research area focus on classifying texts according to their sentiment polarity, which can be positive, negative or neutral (Pang & Lee, 2008). Therefore, it can be considered a text classification problem, since its goal consists of categorizing texts within classes by means of algorithmic methods.

The earliest researches dealing with sentiment analysis consisted on classifying words or phrases according to semantic issues and date from the late 1990s (Hatzivassiloglou & McKeown, 1997). Linguistic heuristics or pre-selected sets of seed words were used. The results obtained in those works served as the basis for classifying entire documents, considering that the average semantic orientation of the words in a review may be an indicator of whether the text is positive or negative (Turney, 2002). The appearance of WordNet (Miller, 1995) and, in general, of annotated corpora, increased the production in this research area. On one hand, WordNet is useful because it allows knowing the semantic relationships between different words. Therefore, with a reduced set of

polarity words, every word could be labeled as positive, negative or neutral through its relationships. On the other hand, corpora and, in particular, the Treebanks, are very useful. They are corpora with the syntactic structure labeled, and are of great help for training the analyzers in order to label the words automatically.

One of the first works that used the term “sentiment analysis” as we currently know it was that presented in (Das & Chen, 2001), which analyzes messages written in stock boards in order to extract the market sentiment. Currently, many of the works in this area focus on document classification based on the sentiment expressed on it. One of the best known domains is that of reviews (Pang, Lee, & Vaithyanathan, 2002) (Dave, Lawrence, & Pennock, 2003). Review websites are examples of especially useful sources for sentiment analysis, such as, e.g. Epinions (Epinions, 1999). Other application areas in which sentiment analysis can be very useful are:

- Recommendation systems (Tatemura, 2000).
- Flame detection (Spertus, 1997).
- Sensitive content detection for advertising (Jin, Li, Mah, & Tong, 2007).
- Human-computer interaction (Liu, Lieberman, & Selker, 2003). – Business Intelligence (Mishne & Glance, 2006)
- Prediction of hostile or negative sources (Abbasi, 2007).
- Classification of citizens’ opinions on a law before its approval: “eRuleMaking” (Cardie, Farina, Bruce, & Wagner, 2006).
- Broadcasting based on the receiver sentiment (Rogers, 2003).
- Dynamic adaptation of daily tools, such as e-mail (Carro, Ballesteros, Ortigosa, Guardiola, & Soriano, 2012). – Marketing or politics (Feldman, 2013).

In general, accuracy is strongly influenced by the context in which the words are used (Turney, 2002) (Aue & Gamon, 2005) (Engström, 2004). For instance, the sentence

“You must read the book” is positive in a book review but is negative if the review is about films.

Additionally, the position of words in text is an interesting factor to consider, since a word at the end of a sentence can change the polarity completely (Pang et al., 2002). For example the sentence “This book is very addictive, it can be read in one sitting, but I have to admit that it is rubbish” begins with the word “addictive” and the expression “one sitting” which are positive in the context of book reviews, but it finish with the word “rubbish” that is negative. Although the sentence contains two positive tokens against one negative, it should be marked as negative because the final word nullifies all the previous ones.

Another issue to be considered is the presence of figures of speech in the analyzed text. Some of them, such as the irony, can change the whole polarity of a text. Sometimes they are difficult to detect even for a human being, if additional information is not provided (e.g. context). Recent work in natural language processing focuses on the detection of these figures, such as (Reyes, Rosso, & Buscaldi, 2012), that build a training dataset of messages written in Twitter with the hashtag ‘#irony’ in order to set a model with machine-learning techniques.

With respect to the techniques used for sentiment analysis, two main approaches are considered: machine-learning methods and lexicon-based approach. The survey written by Pang and Lee (Pang & Lee, 2008) covers the most popular techniques and approaches.

On one hand, machine-learning methods are used to classify texts. An example of the use of machine-learning techniques in order to classify movie reviews is presented in (Pang et al., 2002). It compares different techniques to classify movie reviews, obtaining 82.9% of accuracy when applying Support Vector Machines (SVM). Generally, it is difficult to obtain better results, due to characteristics of natural language. However, in specific domains, the use of machine learning algorithms for classifying texts according to their sentiment orientation performs well.

On the other hand, the lexicon-based approach consists of analyzing the text grammar and executing a function to give a sentiment score to the text, considering a predefined sentiment lexicon (Turney, 2002) (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011). There exist some sentiment lexicon available, such as SentiWordNet (Esuli & Sebastiani, 2006), but it has been noticed that most of the researches build their own lexicon ad hoc, managing the semantic relationships between words with tools such as WordNet (Miller, 1995), already mentioned above.

The great advantage of the lexicon-based approach is that it is not necessary to have a labeled training set to start classifying texts. This approach tends to get worse results than machine learning approaches in specific domains, but when the domain is less bounded the results are better. This is because the lexicon approach analyzes the text grammar, whereas the machine-learning methods fit the algorithms to the training dataset particularities. As an example, in (Taboada et al., 2011) the authors use a lexicon-based method with six different corpora from different domains and obtained 75–80% accuracy. However, when using machine learning (with a preprocessing phase to summarize movie reviews), 86.4% accuracy was achieved (Pang & Lee, 2004). When comparing these two works, the machine-learning approach gets a better accuracy, but it may suffer overfitting to the training dataset, whereas the lexicon-based approach gets a lower accuracy, although is more robust when considering different domains.

The process is as follows : , a dictionary of words is used, in which each word has its sentiment orientation (positive/negative emotional polarity). Each message is classified by following a number of steps. These steps are as follows:

1. **Preprocessing:** In the first step, message is preprocessed by converting all the words into lower-case. Then detection of the idioms is done and they are joined so as to consider as a unique word. For example, as good as is converted into as-good-as.
2. **Segmentation into sentences:** Then, the message is fragmented into sentences. Dots are considered as only punctuation marks that act as a separator at this step.

As other punctuation marks such as commas or semicolons can be part of the emoticons.

3. **Tokenization 1(partial):** In this step, tokens are extracted from each sentence. In this only white spaces are considered to separate the tokens as other punctuation marks such as semi colon, hyphen can be the part of emoticons.
4. **Emoticon detection:** Next is the detection of emoticons. Consecutive occurrence of symbols is considered as presence of emoticons which are compared with text files containing emoticons, extracted from Wikipedia.
5. **Tokenization 2(complete):** in this step, the final set of tokens are extracted by removing all the punctuation marks such as commas, hyphen etc which are left after the removal of emoticons.
6. **Interjection detection:** this step deals in detecting the interjections. The interjections such as hehehe, lolz etc are marked as positive whereas interjections such as uff (tiredness) are marked as negative. This detection of interjection is implemented by the use of regular expressions. Because interjections are observed as set of repeated letters in the word. Such as long sequence of hehehehehe can be considered as strong happy sentiment.
7. **Token score assignation:** the next step is the assignment of the score to each token. 1 is assigned if the token represents positive polarity, -1 is assigned if the token represent the negative polarity; and 0 is assigned if the token is having neutral polarity. To assign a score, the classifier checks if the token is positive/negative emoticons, positive/negative interjection, or whether the word is present in the predefined dictionary. Also repetitive letters are removed in this step if the word does not have any match in the predefined dictionary. As the language written in the social networking sites such as face book is very casual. As in greaaaaat repetition of a leads to undetected word. So removal of a leads to match of the word. Sometimes spelling mistake also lead to the failure of detection of the word, so spelling checker is also used so that word can be corrected and accordingly polarity can be assigned.

8. **Syntactical analysis:** in this stage, syntactical analysis is done, where each token is checked for whether its polarity can be reversed or not which is because of negations. Negations are detected and polarity is reversed for that token.
9. **Polarity calculation:** For calculating polarity of a sentence, the numbers of tokens that are considered for conveying sentiment after the removal of determinant, articles, prepositions etc. are calculated. Such words are considered as stop words. Then each token is assigned a score and polarity score for the sentiment of a sentence is calculated as the sum of the scores divided by the sum of all the candidates to receive a score. The score will lie between -1 to +1.

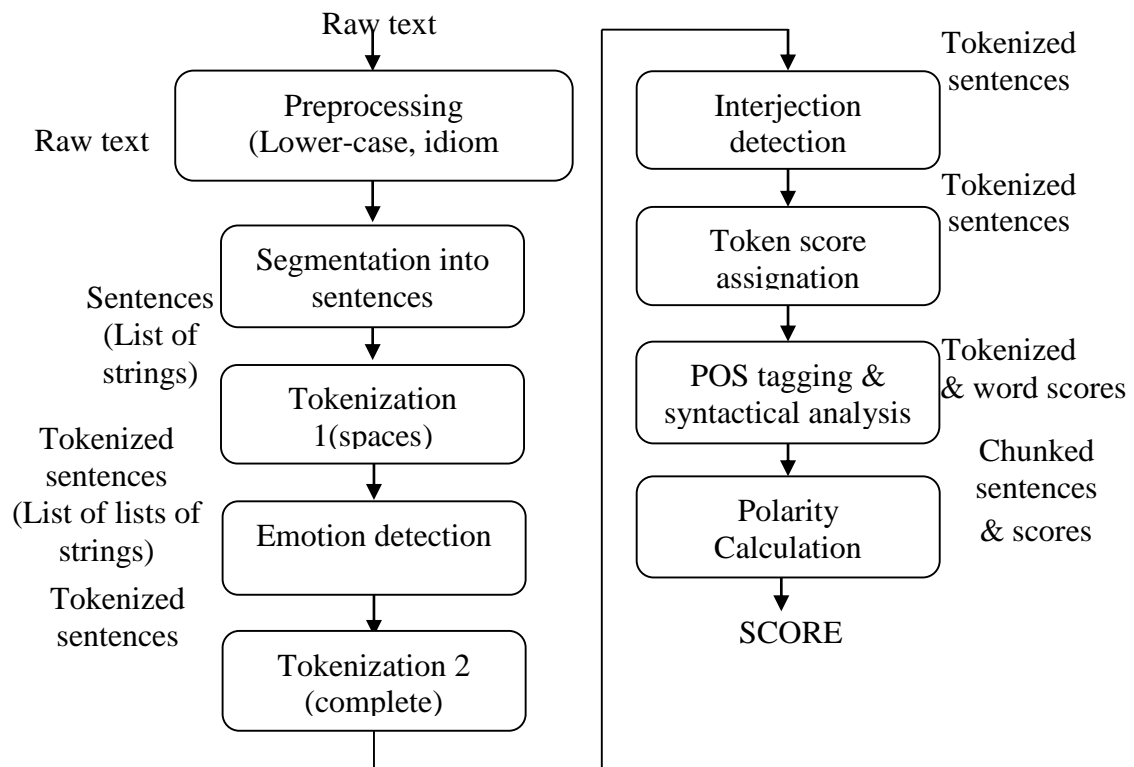


Figure 2 : Flow Chart of Procedure

In relation to language analysis, there are very few works dealing with texts in languages different from English. The works found are usually adaptations of already presented methods for English sentiment analysis. For example, in Martínez Cámara,

Perea, Valdivia, and Ureña (2011), different machine learning methods are applied in order to classify movie reviews, achieving an interesting 86.84% success with SVM in Spanish language.

Finally, in recent years, due to the increasing amount of information delivered through social networks, many researches are focusing on applying sentiment analysis to these data (Go, Bhayani, & Huang, 2009) (Pak & Paroubek, 2010). However, most all these works deal with English texts and retrieve them from Twitter, since it is easier to retrieve data from this social network than from others such as Facebook.

2.2 Classification Levels of sentiment analysis

Sentiment analysis is also known as opinion mining. Sentiment analysis is a natural language processing and information extraction task that aims to obtain writers feelings expressed in positive or negative comments, questions by analyzing a large number of documents.

An opinion is a quadruple (g, s, h, t)

Where g is the opinion (or sentiment) target, s is the sentiment about the target, h is the opinion holder and t is the time when the opinion was expressed.

Sentiment analysis is an ongoing field of research in text mining field. SA is the computational study of Opinions, sentiments, subjectivity toward an entity. The entity can represent individuals, events or topics. The two expressions sentiment analysis and opinion mining are interchangeable.

They express a mutual meaning. But also in some contexts they have different meaning. Opinion mining extracts and analyzes people's opinion about an entity while sentiment analysis identifies the sentiment expressed in a text then analyzes it. Therefore, the target of sentiment analysis is to find opinions, identify the sentiments they express, and then classify their polarity. Sentiment analysis can be considered as a classification process.

Sentiment analysis is considered as a classification process. There are main three classification levels in sentiment analysis:

- Document Level
- Sentence Level
- Aspect Level

- 1. Document Level:** Document Level Sentiment Analysis aims to classify an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit (talking about one topic).
- 2. Sentence Level:** Sentence Level aims to classify sentiment expressed in each sentence. The first step is to identify whether the sentence is subjective or objective. If the sentence is subjective, the sentence level sentiment analysis will determine whether the sentence expresses positive or negative opinions. Sentiment expressions are not necessarily subjective in nature. However, there is no fundamental difference between document and sentence level classifications because sentences are just short documents. Classifying text at the document level or at the sentence level does not provide the necessary detail which is needed in many applications, to obtain these details; we need to go to the aspect level.
- 3. Aspect Level:** Aspect level sentiment analysis aims to classify the sentiment with respect to the specific aspects of entities. The first step is to identify the entity and their aspects. The opinion holder can give different opinions for different aspects of the same entity like this sentence: *“The voice quality of this phone is not good, but the battery life is long”*.

2.3 Serendio: Simple and Practical lexicon based approach to Sentiment Analysis

Abstract: In this paper a lexicon based approach for discovering sentiments is used. Lexicon is built from the Serendio taxonomy which consists of positive, negative, negation, stop words and phrases. A typical tweet contains word variations, emoticons, hashtags etc. Some processing steps such as stemming, emoticon detection and normalization,

exaggerated word shortening and hashtag detection are used in the whole process. After the preprocessing, the lexicon-based system classifies the tweets as positive or negative based on contextual sentiment orientation of the words.

Introduction: Social media websites like Twitter, Facebook etc. are a major hub for users to express their opinions online. On these social media sites, users post comments and opinions on various topics. Hence these sites become rich sources of information to mine for opinions and analyze user behavior and provide insights for: User behaviors, Product feedback, User intentions, Lead generation.

Approach: Serendio Sentiment engine Extracts and analyzes sentiments for a given product and feature set. Serendio sentiment engine currently works for eight different domains such as banking, tablets, smartphones, televisions, apparel, gaming, automobiles and e-readers. The lexicon is manually created in this approach. Two types of lexicons are created.

Common lexicon: this contains data that would have the same semantic meaning or sense across different domains and categories.

- **Common or default sentiment word:** positive and negative words that have the same sentiment value or sense across different domains. For e.g. sentiment word “good” always represents a positive sentiment and it is independent of any category. Positive or negative sentiment words have a sentiment score of +1 or -1 to indicate the respective polarity.
- **Negation Words:** Negation Words are the words which reverse the polarity of sentiment. For example, “the battery life is not good” has negative sentiment.
- **Blind Negation words:** in the sentence, “The T.V needs a better remote”, “needs” is a blind negation word. Blind negation words operate at a sentence level and points out the absence or presence of some sense that is not desired in a product feature.
- **Split words:** Split words are the words used for splitting sentences into clauses. The split words list consists of conjunctions and punctuation marks. For example

the complex sentence,” Camera is good but the battery is bad” is split into two clauses “Camera is good” and “Battery is bad”.

Category specific lexicon: Category specific lexicon contains the (1) Product Catalog which identifies all the products that we are interested in. (2) Feature Catalog which is a list of attributes that the product has. This enables the Serendio engine to do analysis at the feature level. (3) Sentiment words (Positive and negative) that are specific to the category. For example, for a category such as Televisions, a product would be Samsung TV. The feature would be LCD screen and the word “glare” would be the category specific negative sentiment word.

Preprocessing: A typical tweet contains word variations, emoticons, hashtags etc. The objective of the preprocessing step is to normalize the text into an appropriate form to extract the sentiments. Below are the preprocessing steps used.

- POS tagging
- Stemming
- Exaggerated word shortening
- Emoticon detection
- Hashtag Detection

The following algorithm is used in this paper.

```

Algorithm 1: Sentiment Calculation
Data: Preprocessed Twitter data
Result: Output: Positive, Negative, Neutral
Find the list of sentiment words SentiList, its
position in the sentence;
Find the list of sentiment negation words
SentiNegat, its position in the sentence;
Find the list of blind negation words
BlindNegat, its position in the sentence;
if BlindNegat then
| return negativity;
else
| if SentiList and SentiNegat then
| | foreach word in the SentiList do
| | | if word is atmost the distance of 2
| | | | from SentiNegat then
| | | | | Revert the polarity of the word;
| | | | end
| | | end
| | else
| | | if SentiNegat then
| | | | Add the SentiNegat to the
| | | | negative SentiList;
| | | | end
| | | end
| | end
end
SentiSum=0;
foreach word in the SentiList do
| SentiSum=SentiSum+sentiment of
| word;
end
if Hashtag is present then
| Find all the sentiment words in hash tag
| using regex matching and add them to
| SentiList
end
if Emoticon is present then
| Find sentiment of the emoticon and add
| emoticon, it's sentiment to SentiList
end
SentiT ype="neutral";
if SentiSum > 0 then
| SentiT ype="positive";
end
if SentiSum < 0 then
| SentiT ype="negative";
end
return SentiT ype;

```

Figure 3 : Algorithm used for preprocessing

2.4 Method Proposed

In this work, it is proposed to obtain information about user emotional state by obtaining the messages written by them on twitter and applying sentiment analysis to them. The information. The emotion classifier will follow a lexicon-based approach. The outcome of the lexicon classifier will be that it will show classify the tweets as positive/negative/neutral.

2.4.1 Extracting posts from Twitter

The programming language that has been used is Python, for extracting posts from twitter.

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports the following programming paradigms :

- **Object Oriented Programming** : Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated. In object-oriented programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object-oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.
- **Imperative** : Imperative programming is focused on describing how a program operates. In computer science terminologies, imperative programming is a

programming paradigm that describes computation in terms of statements that change a program state. In much the same way that the imperative mood in natural languages expresses commands to take action, imperative programs define sequences of commands for the computer to perform. The term is often used in contrast to declarative programming, which focuses on what the program should accomplish without prescribing how to do it in terms of sequences of actions to be taken.

- **Functional Programming** : In computer science, functional programming is a programming paradigm, a style of building the structure and elements of computer programs, that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. It is a declarative programming paradigm, which means programming is done with expressions. In functional code, the output value of a function depends only on the arguments that are input to the function, so calling a function f twice with the same value for an argument x will produce the same result $f(x)$ each time. Eliminating side effects, i.e. changes in state that do not depend on the function inputs, can make it much easier to understand and predict the behavior of a program, which is one of the key motivations for the development of functional programming. Functional programming has its roots in *lambda calculus*, a formal system developed in the 1930s to investigate computability, the Entscheidungs problem, function definition, function application, and recursion. Many functional programming languages can be viewed as elaborations on the lambda calculus. In the other well-known declarative programming paradigm, logic programming, relations are at the base of respective languages.
- **Procedural Programming** : Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call. Procedures, also known as routines, subroutines, methods, or functions (not to be confused with mathematical functions, but similar to those used in functional programming), simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself. Procedural programming is a list or set of instructions telling a computer what to do step by step and how to perform from the

first code to the second code. Procedural programming languages include C, Go, Fortran, Pascal, and BASIC.

|  | |
|--|--|
| <u>Paradigm(s)</u> | Multi-paradigm: <u>object-oriented</u> , <u>imperative</u> , <u>functional</u> , <u>procedural</u> , <u>reflective</u> |
| <u>Designed by</u> | <u>Guido van Rossum</u> |
| <u>Developer</u> | <u>Python Software Foundation</u> |
| <u>Appeared in</u> | 1991; 23 years ago |
| <u>Stable release</u> | 3.4.2 / 6 October 2014 ^[1] 2.7.9 / 10 December 2014 ^[2] |
| <u>Preview release</u> | 3.4.2 rc1 / 22 September 2014 ^[3] 2.7.9 rc1 / 26 November 2014 ^[4] |
| <u>Typing discipline</u> | <u>duck</u> , <u>dynamic</u> , <u>strong</u> |
| <u>Major implementations</u> | <u>CPython</u> , <u>PyPy</u> , <u>IronPython</u> , <u>Jython</u> |
| <u>Dialects</u> | <u>Cython</u> , <u>RPython</u> , <u>Stackless Python</u> |
| <u>Influenced by</u> | <u>ABC</u> , <u>ALGOL 68</u> , <u>C</u> , <u>C++</u> , <u>Dylan</u> , <u>Haskell</u> , <u>Icon</u> , <u>Java</u> , <u>Lisp</u> , <u>Modula-3</u> , <u>Perl</u> |
| <u>Influenced</u> | <u>Boo</u> , <u>Cobra</u> , <u>D</u> , <u>F#</u> , <u>Falcon</u> , <u>Go</u> , <u>Groovy</u> , <u>JavaScript</u> , <u>Julia</u> , <u>Ruby</u> , <u>Swift</u> |
| <u>OS</u> | <u>Cross-platform</u> |
| <u>License</u> | <u>Python Software Foundation License</u> |
| <u>Filename extension(s)</u> | <u>.py</u> , <u>.pyw</u> , <u>.pyc</u> , <u>.pyo</u> , <u>.pyd</u> |
| <u>Website</u> | <u>www.python.org</u> |

Table 2 : Python Introduction

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, a large number of standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, pseudorandom

number generators, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.

Some parts of the standard library are covered by specifications (for example, the WSGI implementation `wsgiref` follows PEP 333), but the majority of the modules are not. They are specified by their code, internal documentation, and test suite (if supplied). However, because most of the standard library is cross-platform Python code, there are only a few modules that must be altered or completely rewritten by alternative implementations.

The standard library is not essential to run Python or embed Python within an application. Blender 2.49, for instance, omits most of the standard library.

As of October 2014, the Python Package Index, the official repository of third-party software for Python, contains more than 49,000 packages offering a wide range of functionality, including:

- graphical user interfaces, web frameworks, multimedia, databases, networking and communications
- test frameworks, automation and web scraping, documentation tools, system administration
- scientific computing, text processing, image processing

Since 2008, Python has consistently ranked in the top eight most popular programming languages as measured by the TIOBE Programming Community Index.[17] It is the third most popular language whose grammatical syntax is not predominantly based on C, e.g. C++, C#, Objective-C, Java. Python does borrow heavily, however, from the expression and statement syntax of C, making it easier for C programmers to transition between languages.

An empirical study found scripting languages (such as Python) more productive than conventional languages (such as C and Java) for a programming problem involving string manipulation and search in a dictionary. Memory consumption was often "better than Java and not much worse than C or C++".

Large organizations that make use of Python include Google, Yahoo!, CERN, NASA, and some smaller ones like ILM, and ITA.

Python has a very active developer community that creates many libraries which extend the language and make it easier to use various services. One of the many such libraries is Tweepy. Tweepy is open sourced, hosted on GitHub. GitHub is a code sharing and publishing service and a social networking site for programmers. At the heart of GitHub is Git, an open source project started by Linux creator Linus Torvalds. Matthew McCullough, a trainer at GitHub, explains that Git, like other version control systems, manages and stores revisions of projects. Although it's mostly used for code, McCullough says Git could be used to manage any other type of file, such as Word documents or Final Cut projects.

Some of Git's predecessors, such as CVS and Subversion, have a central "repository" of all the files associated with a project. McCullough explains that when a developer makes changes, those changes are made directly to the central repository. With distributed version control systems like Git, if you want to make a change to a project you copy the whole repository to your own system. You make your changes on your local copy, then you "check in" the changes to the central server. McCullough says this encourages the sharing of more granular changes since you don't have to connect to the server every time you make a change. GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

Tweepy can be installed in many ways. The particular way that has been used to install Tweepy is using *pip*. *pip* is a tool for installing and managing Python packages, such as those found in the Python Package Index. Some properties of *pip* are that :

- All packages are downloaded before installation. Partially-completed installation doesn't occur as a result.
- Care is taken to present useful output on the console.

- The reasons for actions are kept track of. For instance, if a package is being installed, pip keeps track of why that package was required.
- Error messages should be useful.
- The code is relatively concise and cohesive, making it easier to use programmatically.
- Packages don't have to be installed as egg archives, they can be installed flat (while keeping the egg metadata).
- Native support for other version control systems (Git, Mercurial and Bazaar)
- Uninstallation of packages.
- Simple to define fixed sets of requirements and reliably reproduce a set of packages.

Once Tweepy is installed, we are now ready to write down and implement the code for extracting tweets from Twitter and storing them in some file.

2.4.2 Using a Dictionary Based Approach (In Progress)

For using a dictionary based approach on the data that has been extracted, we use the NLTK tool in Python.

NLTK stands for Natural Language Toolkit. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and an active discussion forum.

NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit.

2.4.3 Sentiment analysis algorithms and applications: A survey

Sentiment analysis is considered as Sentiment Classification Problem. The first step in the sentiment classification problem is to extract and select text features. This is the

survey paper which described the sentiment analysis in detail. The survey paper gives brief explanation to famous feature selection and sentiment classification algorithms. Some of the current features are:

- **Term presence and frequency:** These features are individual words or word n-grams and their frequency counts. It either gives the words binary weighting or uses term frequency weights to indicate the relative importance of features.
- **Parts of speech (POS):** finding adjectives, as they are important indicators of opinions.
- **Opinion words and phrases:** these are words commonly used to express opinions including good or bad, like or hate. On the other hand, some phrases express opinions without using opinion words. For example: cost me an arm and a leg.

Negations: the appearance of negative words may change the opinion orientation like not good is equivalent to bad

CHAPTER 3 : METHODOLOGY

To start working with Python, we first need to install the Python interpreter, or simply Python on the computer. The Python interpreter can be downloaded right from the official website of python, www.python.org. On the website, choose the download link appropriate for windows, and a download for a *.msi* file will start. Name of the *.msi* file will depend upon the version of Python being downloaded.

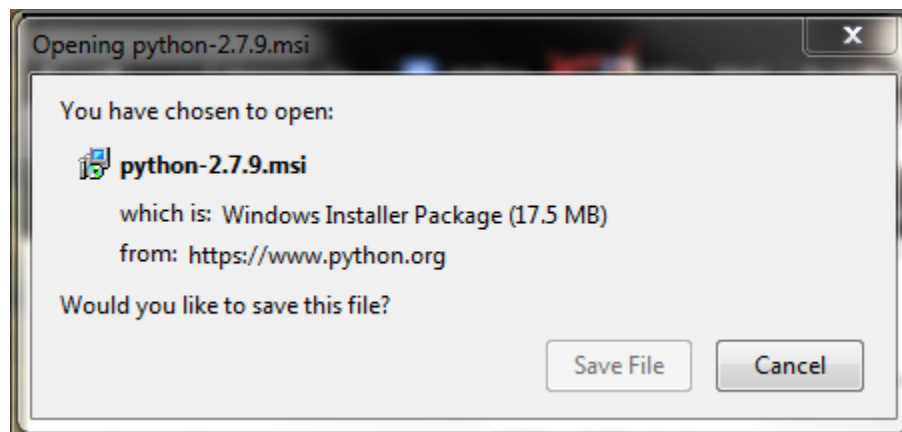


Figure 4 : Downloading Python

Next, we download and install an IDE to start developing in Python. The IDE that has been used here for this project is called **PyCharm**, developed by the Czech company **JetBrains**.

Pip can be installed by downloading *getpip.py*. This file can be found on the following link : <https://pip.pypa.io/en/latest/installing.html> and the file may then be securely downloaded from this link.

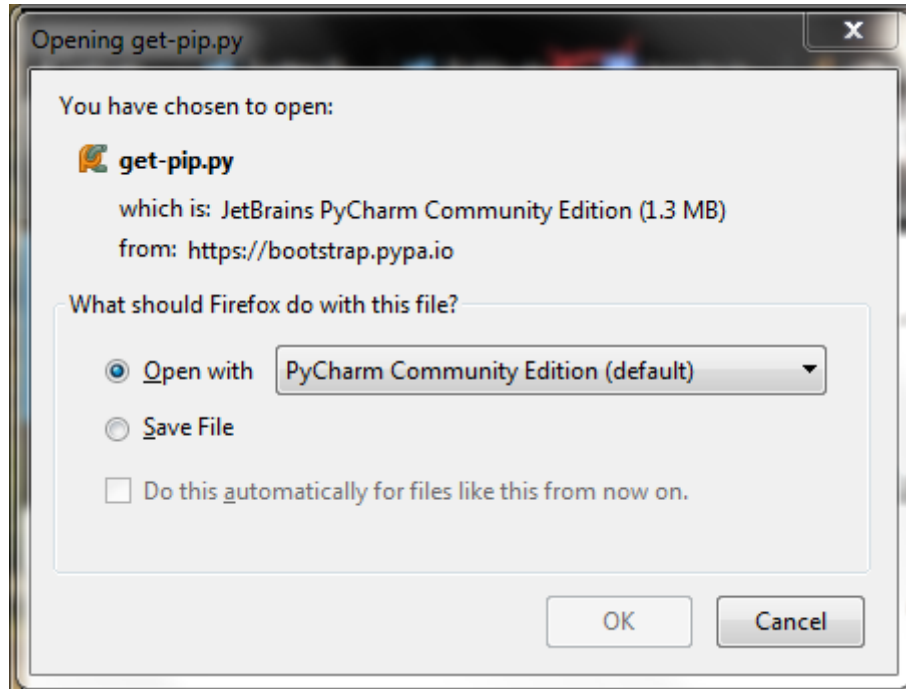


Figure 5 : Getting PIP

Once pip is installed, it will be visible in the Scripts folder of Python

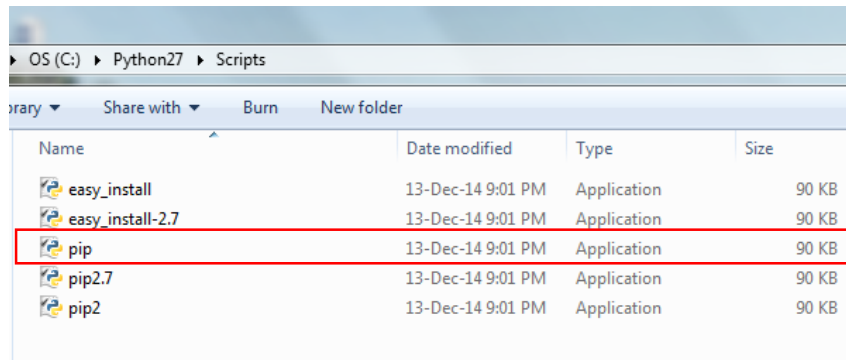


Figure 6 : Locating pip in PythonXX\Scripts

Next, we download Tweepy for installation. Tweepy can be found on <https://github.com/tweepy/tweepy>. From here, we download the zip file and extract the files in some folder.

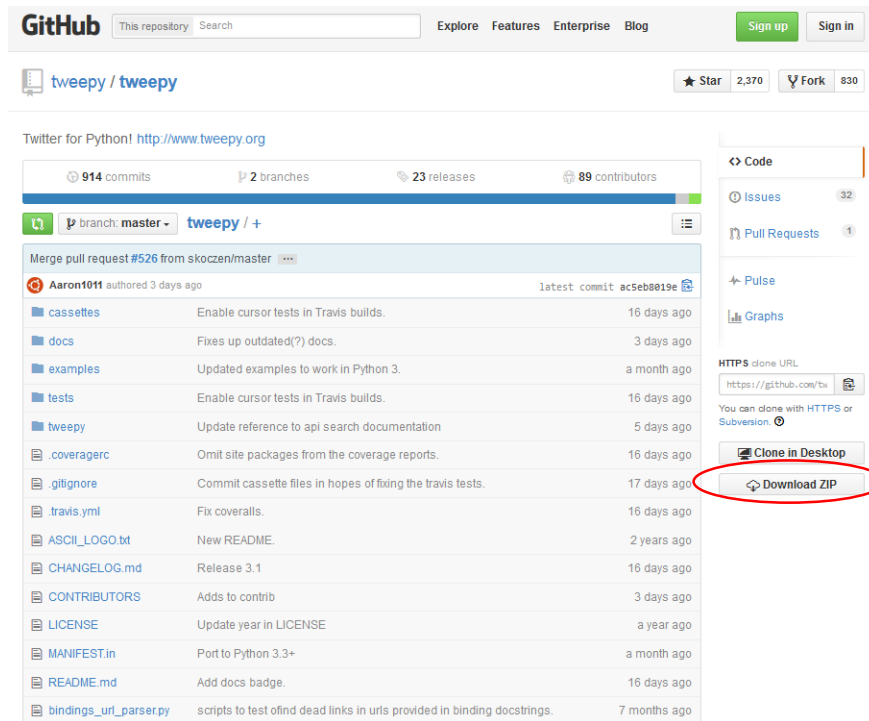


Figure 7 : Downloading Tweepy

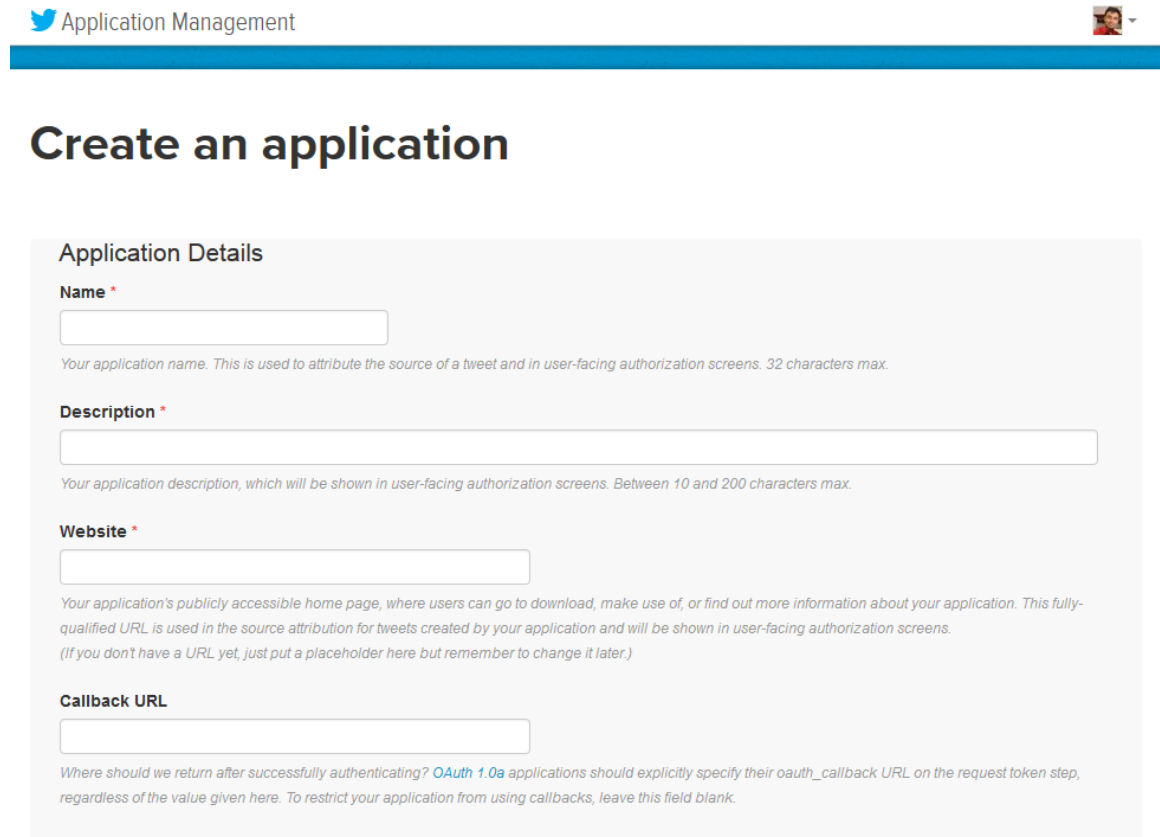
Once Tweepy is downloaded, we now run the command line and perform the installation.

The next step that was performed was creating an App on Twitter and obtaining the Authorization credentials to access Twitter's stream of posts. For this, first step is to go to apps.twitter.com. Please note that a Twitter user account is required before registering ourselves as developers.



Figure 8 : Creating an App

The next page we will view will look like the figure that will follow. It will require the user to fill in the Name & Description of the App and Website of the user registering as a developer.



The screenshot shows the 'Application Management' interface with a blue header bar. The main heading is 'Create an application'. Below it is a form titled 'Application Details' with four sections: 'Name *', 'Description *', 'Website *', and 'Callback URL'. Each section has a text input field and a small explanatory note below it. The 'Name' field is limited to 32 characters. The 'Description' field is limited to 10-200 characters. The 'Website' field is for a fully-qualified URL. The 'Callback URL' field is for the OAuth 1.0a callback URL, which can be left blank to restrict the application from using callbacks.

Figure 9 : Registering and creating a Twitter App

Once this form is filled up, then after the successful creation of the app, the user will be redirected to the following page giving an overview of the details of the App that has just been created.

MyProjectAlpha

Test OAuth

Details Settings Keys and Access Tokens Permissions



Analysing user data
<https://twitter.com/samyakhanda>

Organization

Information about the organization or company associated with your application. This information is optional.

Organization None

Organization website None

Application Settings

Your application's Consumer Key and Secret are used to [authenticate](#) requests to the Twitter Platform.

| | |
|-------------------------|---|
| Access level | Read-only (modify app permissions) |
| Consumer Key (API Key) | FbhHwqKEAt4vZzrqgag1W7hLp (manage keys and access tokens) |
| Callback URL | None |
| Sign in with Twitter | No |
| App-only authentication | https://api.twitter.com/oauth2/token |
| Request token URL | https://api.twitter.com/oauth/request_token |
| Authorize URL | https://api.twitter.com/oauth/authorize |

Figure 10 : Twitter App Drawer

From the above page, the user now needs to create access tokens, which can be created by going to the **Keys and Access Tokens** tab. The Consumer Key and Secret are already present there. All that was needed to do was to generate the Access Token and the Access Token Secret which can be done by pressing the “**Generate My Access Token and Token Secret**” Button.

| | |
|--------------|--|
| Access Level | Read-only (modify app permissions) |
| Owner | samyakhanda |
| Owner ID | [REDACTED] |

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

| | |
|---------------------|-------------|
| Access Token | [REDACTED] |
| Access Token Secret | [REDACTED] |
| Access Level | Read-only |
| Owner | samyakhanda |
| Owner ID | [REDACTED] |

Token Actions

[Regenerate My Access Token and Token Secret](#) [Revoke Token Access](#)

Figure 11 : Generating Access Token and Access Token Secret

Since the above figure was captured from an existing app, therefore currently the button asks to “**Regenerate My Access Token and Token Secret**”. Once all the conditions are met, then we click on the “**Test Oauth**” button which takes us to the following

OAuth Tool

OAuth Settings

Consumer key: *

Consumer secret: *

Remember this should not be shared.

Access token:

Access token secret:

Request Settings

Request type: *

GET POST DELETE PUT HEAD

Figure 12: The OAuth Tool

From here, we record the four parameters for use in the Python Script for accessing the Twitter stream. Once all this is done, we are all set to proceed with the extraction of the Twitter Stream. Tweepy provides access to the well documented Twitter API. With tweepy, it's possible to get any object and use any method that the official Twitter API offers. One of the main usage cases of tweepy is monitoring for tweets and doing actions when some event happens. Key component of that is the *StreamListener* object, which monitors tweets in real time and catches them. The program in use has a *StreamListener* implemented and the code is set up to use OAuth. The *Stream* object is created, which uses that listener as output. *Stream*, being another important object in tweepy also has many methods and "track" is a list of keywords which will trigger the *StreamListener*.

Now that we have extracted the data, the next thing we need to do is break it into sentences. This part will be handled by NLTK. The installation of NLTK is similar to that of Tweepy. NLTK uses functions like *word_tokenize()* and *pos_tag()* for tokenizing the words and Part of Speech Tagging. *word_tokenize()* simply performs the tokenization process on the input text. *Tokenization*, as defined on Wikipedia, is breaking a stream of

text into words, phrases, symbols, or other meaningful elements called tokens. On the tokens that we obtain in the step above, we apply *Part-of-Speech Tagging*. Part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context—i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph. One crucial thing in POS tagging is to identify named entities. These entities can be words that are not understood by the machine at first, but the machine has to be told what they are. For that, we categorize these names as chunks, first. And then we apply named entity recognition to it, using the function *ne_chunk*, that belongs to the NLTK library.

Now we need to make our word base, which will actually be used to compare the words in a post, paragraph, or document, and hence find the result as to the positive/negative/neutral nature of the data being tested. A sample lexicon has been built up, using Thesaurus.com. Thesaurus is a tool that is used to find synonyms of a word. So if we search for a word that is positive in nature, ideally, the search results will also have positive words. So we make two tables for the synonym based lexicon. In the first table, we insert the search word itself, along with its score (+1 for positive, 0 for neutral, and -1 for negative), and consequently the synonyms of the search word, and the score for the synonyms will be the same as that of the search word. In the second table, we store the words that have already been searched for. So, we make an entry into this table, once we have found the synonyms for a given word and stored in the table of words-values.

One problem with our word base that we build for Thesaurus.com is that even for some positive words like ‘good’, the synonyms listed are sometimes the words that mean the exact opposite, for example, ‘bad’. For that, we can make use of the function *fetchone()* from the sqlite3 library, which will ensure that a word is only inserted in the table if it does not already exist in the table.

After populating our word base, we now proceed to check how accurately this works. For that, we can take one document each of type positive and negative and check what our system predicts based on the word base it has.

So we start by making two arrays – one each for positive and negative words. We fetch the positive and negative words from the word-values table using their numeric scores of +1 and -1 respectively. Then we open a file that we want to check for its sentiment, and set the sentiment counter as 0 initially. as we scan the words of the file, we increment the sentiment counter by a certain number for every positive word that is encountered, and decrement the sentiment counter by a certain amount on encountering a negative word. Note that we may have to toggle the increment amount and decrement amount for positive and negative words to obtain more precise results.

The other thing that can be done to increase our accuracy in the testing of our lexicon is that we populate our word base even further. Currently the word base has more positive words than negative, so the positive data is likely to be tested correctly, whereas the negative data might be tested incorrectly and displayed as positive in the result.

4. CODE AND SCREENSHOTS

4.1 Extracting Data from Twitter

```
__author__ = 'SAMYAK'

from tweepy import Stream

from tweepy import OAuthHandler

from tweepy.streaming import StreamListener

import json

ckey = '*****'

csecret = '*****'

atoken = '*****'

asecret = '*****'
```

```
class listener(StreamListener):

    def on_data(self, data):

        try:

            #print data

            decode = json.loads(data)

            textT = decode["text"] + "\r\n\n"

            print textT

            f = open("E:/twitter.txt", "a+")

            f.write(textT + "\r\n")

            f.close()

            return True

        except Exception:

            print "\n"
```

```
auth = OAuthHandler(ckey, csecret)
```

```
auth.set_access_token(accessToken, accessSecret)
```

```
twitterStream = Stream(auth, listener())
```

```
twitterStream.filter(track=["hobbit"])
```

4.2 Word Tokenization using NumPy and NLTK

senti1.py

```
from numpy import *

import nltk

import re

import time

try:

    def splitParagraphIntoSentences(paragraph):

        #break a paragraph into sentences

        #and return a list

        # to split by multiple characters

        # regular expressions are easiest (and fastest)

        sentenceEnders = re.compile('[.])

        sentenceList = sentenceEnders.split(paragraph)

        return sentenceList

text_file = open("C:\Users\SAMYAK\Desktop\comment.txt","r")

text_file1 = open("C:\Users\SAMYAK\Desktop\comment1.txt","w+")

lines=text_file.readlines()

for item in lines:

    sentences = splitParagraphIntoSentences(item)
```

for s in sentences:

```
#data.__add__(s.strip())
```

```
data=s.strip()
```

```
text_file1.write(data)
```

for liness in data:

```
print liness
```

except Exception,e:

```
print("\nThis is the exception block of senti 1")
```

senti.py

```
execfile('senti1.py')
```

```
import numpy
```

```
import nltk
```

```
#execfile('senti1.py')
```

```
try:
```

```
file_content=open("C:\Users\SAMYAK\Desktop\comment1.txt",'r')
```

```
tokens=nltk.word_tokenize(file_content)
```

```
tagged=nltk.pos_tag(tokens)
```

```
print tagged
```

```
chunkgram=r""""chunk:{<RB\w?>*<VB\w?>*<NNP>}""""
```

```
chunkParser=nltk.RegexpParser(chunkgram)

chunked=chunkParser.parse(tagged)

print chunked

chunked.draw()

except Exception,e:

    print ("\n This is the exception Block of senti")
```

senti2.py

```
execfile('senti.py')

from numpy import *

import nltk

#execfile('senti.py')

try:

    file_content=open("C:\Users\SAMYAK\Desktop\comment1.txt").read()

    tokens=nltk.word_tokenize(file_content)

    tagged=nltk.pos_tag(tokens)

    print tagged

    namedEnt=nltk.ne_chunk(tagged, binary=True)

    namedEnt.draw()

except Exception,e:

    print ("\nThis is the exception block of senti 2")
```

4.3 Building the Word Base

```
__author__ = 'SAMYAK'
```

```
import time
```

```
import urllib2
```

```
from urllib2 import urlopen
```

```
import re
```

```
import cookielib
```

```
from cookielib import CookieJar
```

```
import datetime
```

```
import sqlite3
```

```
import nltk
```

```
cj = CookieJar()
```

```
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
```

```
opener.addheaders = [('User-agent', 'Mozilla/5.0')]
```

```
conn = sqlite3.connect('knowledgeBase.db')
```

```
c = conn.cursor()
```

```
startingWord = 'excited'
```

```
startingWordVal = 1
```

```
synArray = []
```

```
def main():
```

```
    try:
```

```
        page = 'http://thesaurus.com/browse/' + startingWord + '?s=t'
```

```
        sourceCode = opener.open(page).read()
```

```
    try:
```

```
        synoNym = sourceCode.split('<div class="relevancy-list">')
```

```
        x = 1
```

```
        while x < len(synoNym):
```

```
            try:
```

```
                synoNymSplit = synoNym[x].split('<section class="container-info  
antonyms">')[0]
```

```
                synoNyms = re.findall(r'<span class="text">(\w*?)</span>', synoNymSplit)
```

```
                print synoNyms
```



```
for eachSyn in synoNyms:
```

```
    query = "SELECT * FROM wordVals WHERE word =?"
```

```
    c.execute(query, [(eachSyn)])
```

```
    data = c.fetchone()
```

```
    if data is None:
```

```
        print 'not here yet. it is added now.'
```

```
        c.execute("INSERT INTO wordVals (word, value) VALUES (?,?)",
```

```
                (eachSyn, startingWordVal))
```

```
        conn.commit()
```

```
    else:
```

```
        print 'word already here'
```

```
except Exception, e:
```

```
    print str(e)
```

```
    print 'failed try block 3'
```

```
    x += 1
```

```
except Exception, e:
```

```
    print str(e)
```

```
print 'failed try block 2'
```

```
except Exception, e:
```

```
    print str(e)
```

```
    print 'failed in the main loop'
```

```
main()
```

```
c.execute("INSERT INTO doneSyns (word, value) VALUES (?,?)", (startingWord,  
startingWordVal) )
```

```
conn.commit()
```

4.4 Testing the word base on a sample document

```
__author__ = 'SAMYAK'
```

```
import sqlite3
```

```
import time
```

```
conn = sqlite3.connect('knowledgeBase.db')
```

```
c = conn.cursor()
```

```
negativeWords = []
```

```
positiveWords = []
```

```
sql = "SELECT * FROM wordVals WHERE value = ?"
```

```
def loadWordArrays():
```

```
    for negRow in c.execute(sql, [(-1)]):
```

```
        negativeWords.append(negRow[0])
```

```
    print 'negative words loaded'
```

```
    for posRow in c.execute(sql, [(1)]):
```

```
    positiveWords.append(posRow[0])

print 'positive words loaded'

def testSentiment():

    readfile = open('negativeIMDB.txt', 'r').read()

    sentCounter = 0

    for eachPosWord in positiveWords:

        if eachPosWord in readfile:

            print eachPosWord

            sentCounter += .2

    print '*****'

    for eachNegWord in negativeWords:

        if eachNegWord in readfile:

            print eachNegWord

            sentCounter -= 1.3
```

```
if sentCounter > 0:  
    print 'this text is positive'
```

```
if sentCounter == 0:  
    print 'this text is neutral'
```

```
if sentCounter < 0:  
    print 'this text is negative'
```

```
print sentCounter
```

```
loadWordArrays()
```

```
testSentiment()
```

4.5 Screenshots

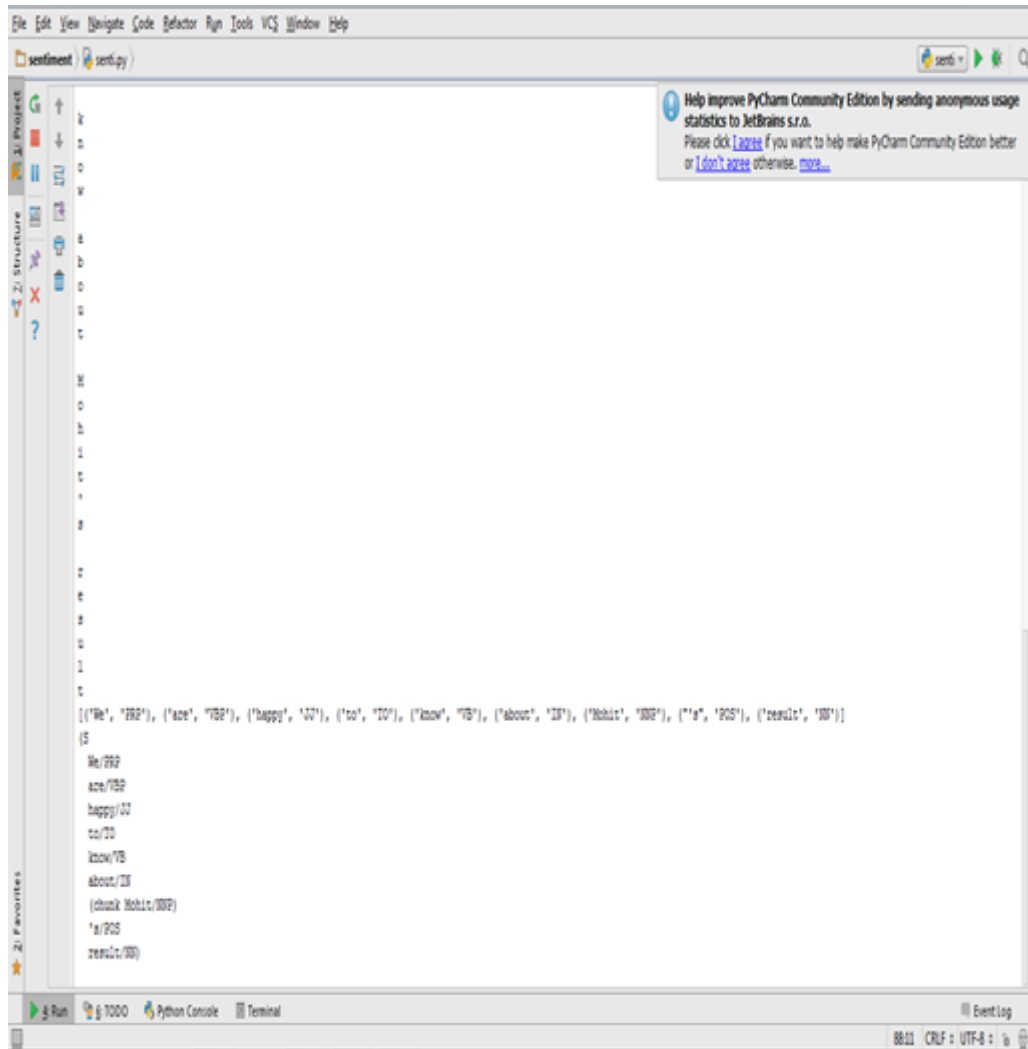


Figure 13 : Text tokenization through Python and NLTK

Task Performed: Dictionary based approach is implemented but not completed

yet

PyCharm IDE and NLTK tools are used. Sub Tasks Performed:

- Convert Document into sentences.
- Sentences into tokens.
- Tag the POS in tokens
- Created a chunked and named Entity Tree of POS parts.

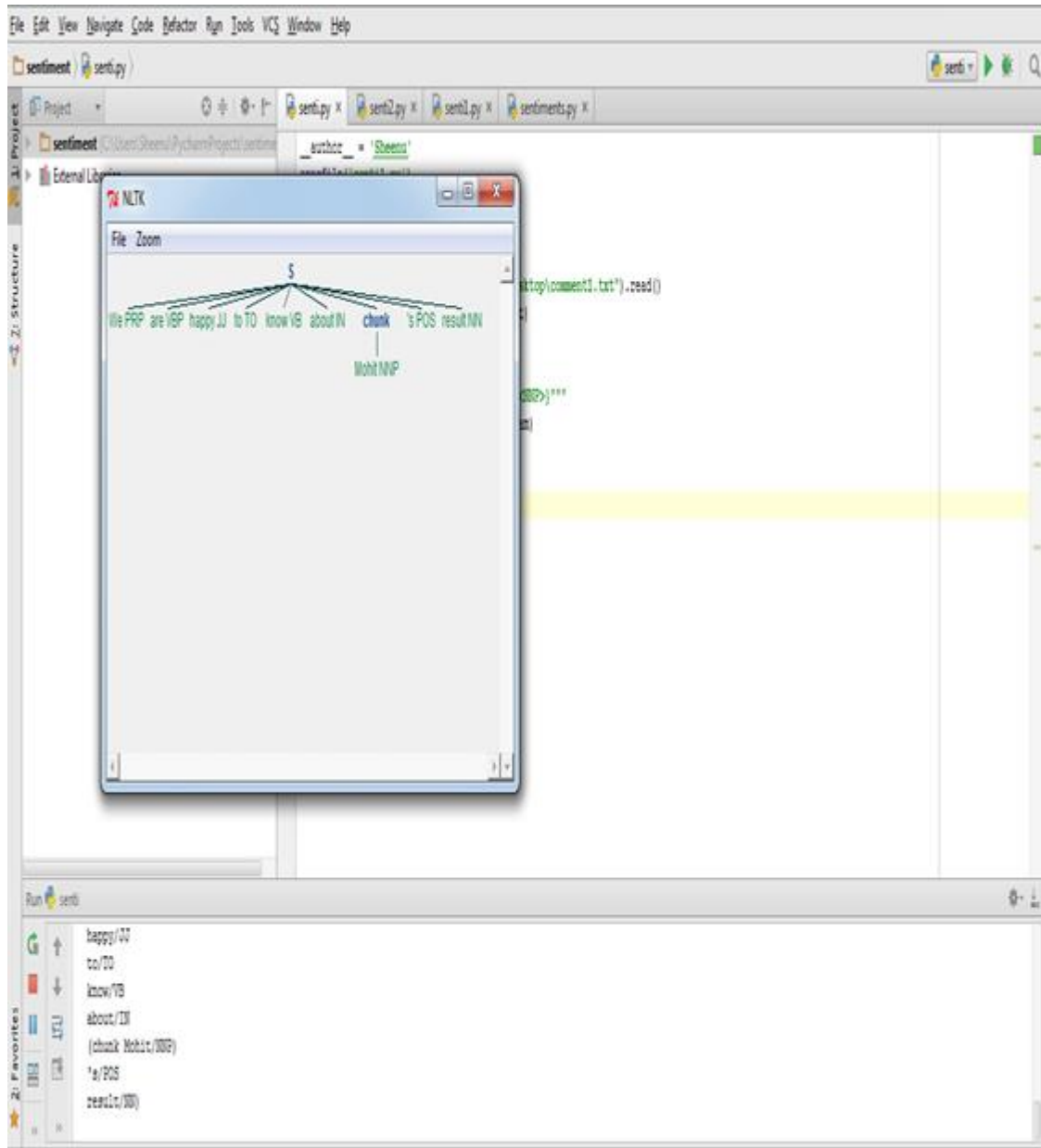


Figure 14 : Creating the chunked tree presenting POS and named entity


```
Run senTest
here
anything
script
benchmark
ever
*****
bad
off
rough
wicked
ill
dry
out
rot
little
port
hole
mar
opposition
loser
imply
rip
fault
sin
mortal
problem
bound
confuse
rouse
pose
will
ask
act
hat
car
no
this text is positive
1.8

Process finished with exit code 0
```

Figure 16 : Analyzing a document for sentiment

CHAPTER 5 : CONCLUSION

5.1 Conclusion

So far, during the course of one academic year, as a part of the project, we have seen that quite a few tasks have been performed, like Extraction of Data from Twitter using python programming, where first we have made a developer account with twitter to gain access tokens to access the twitter data stream and then writing a python program to retrieve the data stream from twitter and store it in a text file. The next thing we did is tokenization, part of speech tagging and named entity recognition using NLTK, where we input a stream of sentences and break it into tokens, identify the parts of speech and the names in the sentences. Then we move on to building our word base in which we store words and their corresponding polarities. And then we finally take a document and check its sentiment using the word base that we build and populate. An attempt is being made to refine the lexicon that has been made to obtain more accurate results and determine an appropriate number by which the sentiment counter should be incremented in case of a positive word and another appropriate number by which the sentiment counter should be decremented to in case of a negative number.

REFERENCES

- [1] Jose M. Martin, Alvaro Ortigosa, Rosa M. Carro, “*SentBuk: Sentiment analysis for e-learning environments*”, IEEE Conference, Computers in Education (SIIE), 2012, pp. 1-6.
- [2] Alvaro Ortigosa *, Jose M. Martin, Rosa M. Carro, “*Sentiment analysis in Facebook and its application to e-learning*”, ELSEVIER, Journal of Computers in Human Behavior, 2013, pp 1-15.
- [3] Prabu Palanisamy, Vineet Yadav and Harsha Elchuri, “*Serendio: Simple and Practical lexicon based approach to Sentiment Analysis*”, Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International workshop on Semantic Evaluation (SemEval 2013), June 14-15, 2013, pp. 543-548.
- [4] Walaa Medhat, Ahmed Hassan, Hoda Korashy, “*Sentiment analysis algorithms and applications: A survey*”, ELSEVIER Ain Shams Engineering Journal, 2014, pp. 1093-1113.
- [5] Ronen Feldman, “*Techniques and Applications for Sentiment Analysis*”, Communications of the ACM, Vol. 56, No.4, April 2013, pp. 82-89.
- [6] Xiaowen Ding Chicago, Bing Liu, Philip S. YU, “*A holistic lexicon-based approach to opinion mining*” proceedings of the 2008 International Conference on Web Search and Data Mining, 2008, pp. 231-240.
- [7] http://en.wikipedia.org/wiki/Python_%28programming_language%29
- [8] http://en.wikipedia.org/wiki/Object-oriented_programming
- [9] http://en.wikipedia.org/wiki/Imperative_programming
- [10] http://en.wikipedia.org/wiki/Functional_programming
- [11] <http://www.pythoncentral.io/introduction-to-tweepy-twitter-for-python/>
- [12] <http://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/>
- [13] <https://pip.pypa.io/en/1.0.2/>
- [14] <http://www.nltk.org/>
- [15] http://en.wikipedia.org/wiki/Natural_Language_Toolkit
- [16] Jiawei Han and Micheline Kamber, “*Graph Mining, Social Network Analysis, and Multirelational Data Mining*”, Data Mining Concepts and Techniques, 2nd edition, Morgan Kauffman ,555-556