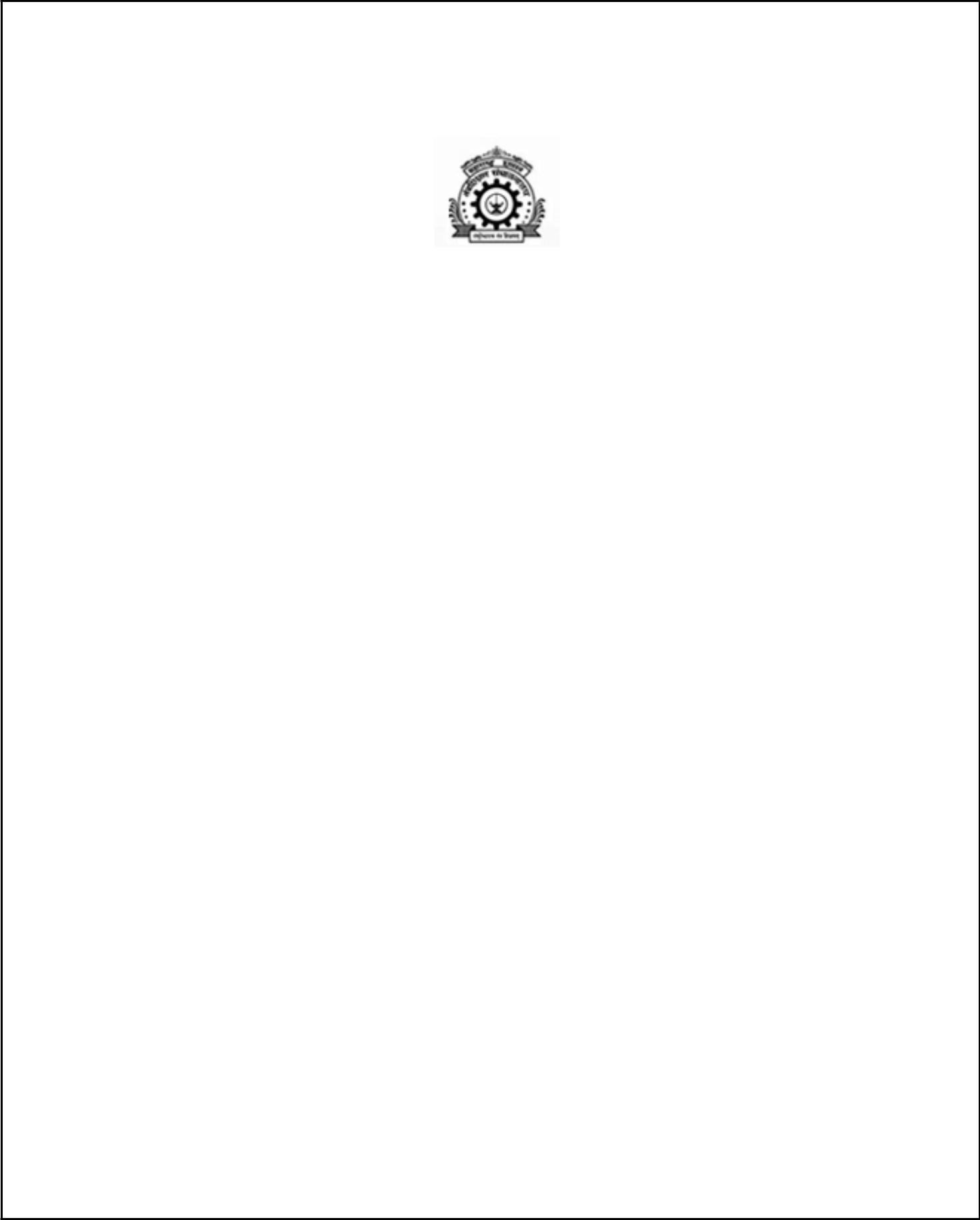
***Government Polytechnic,Solapur Department of Computer Technology***

***CERTIFICATE***

*It is to certify that*

*Miss.Nadargi V.R.*

*Miss.Chendake P.M.*

*Miss.Chinni P.A.*

*Miss.Kadadas K.S.*

*Class:CM6G*

*has satisfactorily completed the project titled as*

*Sentiment Analysis*

*as a partial fulfi****l****ment*

*for awarding the Diploma in ComputerTechnology*

*by Maharashtra State Board of Technical Education,Mumbai*

*for the year2017-2018.*

*Project Guide Principal*

*(Gangundi R.G)* *Head of Department*  *(Katare D.A.)*

*(Tarange A.L.)*

*Internal Examiner* *External*

*Examiner*

*Date:*

*Place:Solapur*

**ACKNOWLEDGEMENTS**

We are deeply thankful to our principal Mr.D.A.Katare, head of department Mr.A.L.Tarange Sir and our guide Prof.Ravindra Gangundi Sir for helping us in accomplishing our final project. Their guidance, support and motivation enabled us in achieving the objectives of the project.

**INDEX**

CERTIFICATE................................................................................................................01

ACKNOWLEDGEMENTS............................................................................................ 02

ABSTRACT.................................................................................................................... 05

INTRODUCTION .......................................................................................................... 06

* Domain Introduction
* Positive
* Negative
* Neutral
* Feature Extraction
* Tokenization

CHAPTER 1 :- Collecting the data.................................................................................10

* Register your twitter account
* Steps for registrating to your twitter API
* Accessing the data
* What is JSON?

CHAPTER 2 :- Sentiment Analysis................................................................................12

* What is sentiment analysis?
* Twitter sentiment analysis
* Previous work
* Why sentiment analysis?

CHAPTER 3 :- Text Preprocessing.................................................................................13

* What is text preprocessing?
* The anatomy of tweet
* How to tokenize a tweet text?

CHAPTER 4 :- Data Visualization..................................................................................16

* Data visualization: a wise investment in your big data future
* Why is data visualization important?
* Matplotlib
* NumPy

CHAPTER 5:- Navie Bayes Algorithm...........................................................................20

* What is Navie Bayes Algorithm?
* Pros
* Cons
* Applications of Navie Bayes Algorithm

LIST OF DIAGRAMS......................................................................................................22

* Use-Case Diagram
* Sequence Diagram

CONCLUSION.................................................................................................................24

REFERENCES .................................................................................................................25

**ABSTRACT**

This project addresses the problem of sentiment analysis in twitter, that is classifying tweets according to the sentiment expressed in them : positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users out of which 100 million are active users and half of them log on twitter on a daily basis generating nearly 250 million tweets per day.

Due to this large amount of usage we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.

**INTRODUCTION**

Motivation We have chosen to work with twitter since we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompt and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis).

Sentiment analysis of public is highly critical in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could be done by analysing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm’s stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favourable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets for particular locations. If firms can get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions like creating suitable market segments. Predicting the results of popular political elections and polls is also an emerging application to sentiment analysis.

**Domain Introduction:**

This project of analyzing sentiments of tweets comes under the domain of and “Data Mining”. they can be formally defined as the process of discovering “useful” patterns in large set of data, either automatically (unsupervised) or semiautomatically (supervised). The project would heavily rely on techniques of “Natural Language Processing” in extracting significant patterns and features from the large data set of tweets and on “Machine Learning” techniques for accurately classifying individual unlabelled data samples (tweets) according to whichever pattern model best describes them.

Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word “excellent” has a strong positive connotation while the word “evil” possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analyzing the frequency distribution of these parts of speech (ether individually or collectively with some other part of speech) in a particular class of labeled tweets. Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word .

sentiment analysis and has been used by various researchers. The simplest way to incorporate this model in our classifier is by using unigrams as features. Generally speaking n-grams is a contiguous sequence of “n” words in our text, which is completely independent of any other words or grams in the text. So unigrams is just a collection of individual words in the text to be classified, and we assume that the probability of occurrence of one word will not be affected by the presence or absence of any other word in the text.

This is a very simplifying assumption but it has been shown to provide rather good performance. One simple way to use unigrams as features is to assign them with a certain prior polarity, and take the average of the overall polarity of the text, where the overall polarity of the text could simply be calculated by summing the prior polarities of individual unigrams. Prior polarity of the word would be positive if the word is generally used as an indication of positivity, for example the word “sweet”; while it would be negative if the word is generally associated with negative connotations, for example “evil”. There can also be degrees of polarity in the model, which means how much indicative is that word for that particular class. A word like “awesome” would probably have strong subjective polarity along with positivity, while the word “decent” would although have positive prior polarity but probably with weak subjectivity.

There are three ways of using prior polarity of words as features. The simpler un-supervised approach is to use publicly available online lexicons/dictionaries which map a word to its prior polarity. The Multi-Perspective-Question-Answering (MPQA) is an online resource with such a subjectivity lexicon which maps a total of 4,850 words according to whether they are “positive” or “negative” and whether they have “strong” or “weak” subjectivity . The SentiWordNet 3.0 is another such resource which gives probability of each word belonging to positive, negative and neutral classes . The second approach is to construct a custom prior polarity dictionary from our training data according to the occurrence of each word in each particular class. For example if a certain word is occurring more often in the positive labelled phrases in our training dataset (as compared to other classes) then we can calculate probability of that word belonging to positive class to be higher than the probability of occurring in any other class. This approach has been shown to give better performance, since the prior polarity of words is more suited and fitted to a particular type of text and is not very general like in the former approach. However, the latter is a supervised approach because the training data has to be labelled in the appropriate classes before it is possible to calculate the relative occurrence of a word in each of the class.

The third approach is a middle ground between the above two approaches. In this approach we construct our own polarity lexicon but not necessarily from our training data, so we don’t need to have labelled training data. One way of doing this as proposed by Turney et al. is to calculate the prior semantic orientation (polarity) of a word or phrase by calculating it’s mutual information with the word “excellent” and subtracting the result with the mutual information of that word or phrase with the word “poor” .

Grammatical features (like “Parts of Speech Tagging” or POS tagging) are also commonly used in this domain. The concept is to tag each word of the tweet in terms of what part of speech it belongs to: noun, pronoun, verb, adjective, adverb, interjections, intensifiers etc. The concept is to detect patterns based on these POS and use them in the classification process. For example it has been reported that objective tweets contain more common nouns and third-person verbs than subjective tweets so if a tweet to be classified has a proportionally large usage of common nouns and verbs in third person, that tweet would have a greater probability of being objective (according to this particular feature). Similarly subjective tweets contain more adverbs, adjectives and interjections

***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: “I am feeling happy”.

***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: “She got failure in life ”.

***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: “Nobody gets the day off”

***Feature Extraction:***

Now that we have arrived at our training set we need to extract useful features from it which can be used in the process of classification. But first we will discuss some text formatting techniques which will aid us in feature extraction:

***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.

• Url’s and user references (identified by tokens “http” and “@”) are removed if we are interested in only analyzing the text of the tweet.

• Punctuation marks and digits/numerals may be removed if for example we wish to compare the tweet to a list of English words.

• Lowercase Conversion: Tweet may be normalized by converting it to lowercase which makes it’s comparison with an English dictionary easier.

• Stemming: It is the text normalizing process of reducing a derived word to its root or stem. For example a stemmer would reduce the phrases “stemmer”, “stemmed”, “stemming” to the root word “stem”. Advantage of stemming is that it makes comparison between words simpler, as we do not need to deal with complex grammatical transformations of the word.

• Stop-words removal: Stop words are class of some extremely common words which hold no additional information when used in a text and are thus claimed to be useless . Examples include “a”, “an”, “the”, “he”, “she”, “by”, “on”, etc. It is sometimes convenient to remove these words because they hold no additional information since they are used almost equally in all classes of text, for example when computing prior-sentiment-polarity of words in a tweet according to their frequency of occurrence in different classes and using this polarity to calculate the average sentiment of the tweet over the set of words used in that tweet.

• Parts-of-Speech Tagging: 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.

**Chapter 1: Collecting the data**

Twitter is a popular social network where users can share short SMS-like messages called tweets. Users share thoughts, links and pictures on Twitter, journalists comment on live events, companies promote products and engage with customers. The list of different ways to use Twitter could be really long, and with 500 millions of tweets per day, there’s a lot of data to analyse and to play with.

***Register Your twitter account***

***Steps for registering to your twitter API***

* In order to have access to Twitter data programmatically, we need to create an app that interacts with the Twitter API
* The first step is the registration of your app. In particular, you need to point your browser to http://apps.twitter.com
* log-in to Twitter (if you’re not already logged in) and register a new application. You can now choose a name and a description for your app (for example “Mining Demo” or similar).
* You will receive a consumer key and a consumer secret: these are application settings that should always be kept private.
* From the configuration page of your app, you can also require an access token and an access token secret.
* Similarly to the consumer keys, these strings must also be kept private: they provide the application access to Twitter on behalf of your account.
* The default permissions are read-only, which is all we need in our case, but if you decide to change your permission to provide writing features in your app, you must negotiate a new access token[4][5].

***Accessing the data***

Twitter provides REST APIs you can use to interact with their service. There is also a bunch of Python-based clients out there that we can use without re-inventing the wheel. In particular, Tweepy in one of the most interesting and straightforward to use, so let’s install it:

*pip install tweepy==3.3.0*

In order to authorise our app to access Twitter on our behalf, we need to use the OAuth interface:

*import tweepy*

*from tweepy import OAuthHandler*

*consumer\_key = 'YOUR-CONSUMER-KEY'*

*consumer\_secret = 'YOUR-CONSUMER-SECRET'*

*access\_token = 'YOUR-ACCESS-TOKEN'*

*access\_secret = 'YOUR-ACCESS-SECRET'*

*auth = OAuthHandler(consumer\_key, consumer\_secret)*

*auth.set\_access\_token(access\_token, access\_secret)*

*api = tweepy.API(auth)*

The api variable is now our entry point for most of the operations we can perform with Twitter.

For example, we can read our own timeline (i.e. our Twitter homepage) with:

*for status in tweepy.Cursor(api.home\_timeline).items(10):*

# Process a single status

*print(status.text)*

Tweepy provides the convenient Cursor interface to iterate through different types of objects. In the example above we’re using 10 to limit the number of tweets we’re reading, but we can of course access more. The status variable is an instance of the Status() class, a nice wrapper to access the data. The JSON response from the Twitter API is available in the attribute \_json (with a leading underscore), which is not the raw JSON string, but a dictionary[4].

*for status in tweepy.Cursor(api.home\_timeline).items(10):*

# Process a single status

*process\_or\_store(status.\_json)*

***What is JSON ??***

All Twitter APIs that return Tweets provide that data encoded using JavaScript Object Notation (JSON). JSON is based on key-value pairs, with named attributes and associated values. These attributes, and their state, are used to describe objects.

At Twitter we serve many objects as JSON, including Tweets and Users. These objects all encapsulate core attributes that describe the object. Each Tweet has an author, a message, a unique ID, a timestamp of when it was posted, and sometimes geo metdata shared by the user. Each User has a Twitter name, an ID, a number of followers, and most often an account biodata.

With each Tweet we also generate 'entity' objects, which are arrays of common Tweet contents such as hashtags, mentions, media, and links. If there are links, the JSON payload can also provide metadata such as the fully unwound URL and the webpage’s title and description[7]**.**

**Chapter 2:Sentiment Analysis**

***What is sentiment analysis?*** Sentiment Analysis is the process of ‘computationally’ determining whether a piece of writing is positive, negative or neutral. It’s also known as opinion mining, deriving the opinion or attitude of a speaker.

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Consumers can use sentiment analysis to research products and services before a purchase. Production companies can use the public opinion to determine acceptance of their products and the public demand. Movie-goers can decide whether to watch a movie or not after going through other people’s reviews.

***Twitter Sentiment Analysis:***

Traditionally, most of the research in sentiment analysis has been aimed at larger pieces of text, like movie reviews, or product reviews. Tweets are more casual and are limited by 140 characters. However, this alone does not make it an easy task (in terms of programming time, not in accuracy as larger piece of text tends to be correctly classified) as people rarely give a second thought before posting a tweet. Grammar and content both suffer at the hands of the tweeter. The presence of a large dataset is always recommended (for better training of the classifier) and twitter makes it possible to obtain any number of tweets during a desired period. However, various difficulties are faced during processing of raw tweets.

***Previous Work:***

Alec Go, Richa Bhayani and Lei Huang (Students at Stanford University) have done some serious work in twitter sentiment analysis. Even though their source code is not publicly available, their approach was to use machine learning algorithm for building a classifier, namely Maximum Entropy Classifier.The use of a large dataset too helped them to obtain a high accuracy in their classification of tweets’sentiments. The data set used by them is however public and I too have used the same data set in order to obtain results as close to theirs as possible. Other noteworthy works are by Laurent Luce and Niek Sanders. Both of them used quite smaller datasets, but their work consisted of some insightful approaches.

***Why sentiment analysis?***

* Business:In marketing field companies use it to develop their strategies, to understand customers’ feelings towards products or brand, how people respond to their campaigns or product launches and why consumers don’t buy some products.
* Politics:In political field, it is used to keep track of political view, to detect consistency and inconsistency between statements and actions at the government level. It can be used to predict election results as well!
* Public Actions:Sentiment analysis also is used to monitor and analyse social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere[6].

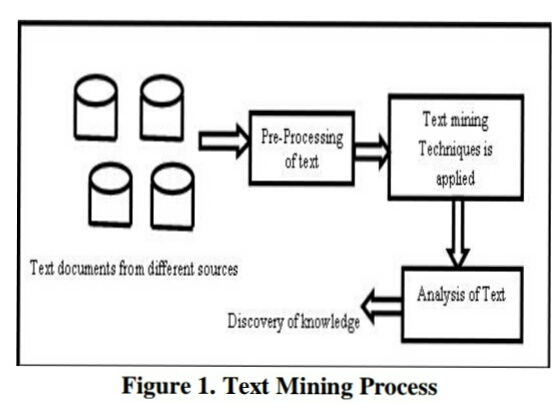
**Chapter 3:Text-Processing**

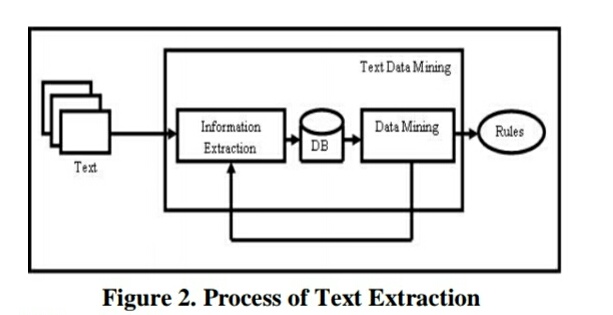
Text mining is the process of seeking or extracting the useful information from the textual data. It is exciting research area as it tries to discoverknowledge from unstructured texts. It is also known as Text Data Mining (TDM) and knowledge Discovery in Textual Databases (KDT). KDT plays an increasingly significant role in emerging applications, such as Text Understanding. Text mining process is same as data mining, except, the data mining tools are designed to handle structured data whereas text mining can able to handleunstructured or semi-structured data sets such as emails HTML files and full text documents etc.. Text Mining is used for finding the new, previously unidentified information from different written resources. Structured data is data that resides in a fixed field within a record or file. This data is contained in relational database and spreadsheets.

The unstructured data usually refers to information that does not reside in a traditional row-column database and it is the opposite of structured data. Semi-Structured data is the data that is neither raw data, nor typed data in a conventional database system. Text mining is a new area of computer science research that tries to solve the issues that occur in the area of data mining, machine learning, information extraction, retrieval, knowledge management and classification .

Categorization involves identifying the main themes of a document by inserting the document into a pre-defined set of topics. When categorizing a document, a computer program will often treat the document as a “bag of words.” It does not try to process the actual information as information extraction does. Rather, the categorization only counts words that appear and, from the counts, identifies the main topics that the document covers.

Categorization often relies on a glossary for which topics are predefined, and relationships are identified by looking for large terms, narrower terms, synonyms, and related terms.Natural Language Processing Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text. NLP researchers aim to collect knowledge on how human beings understand and use language so that fitting tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the preferred tasks.The basics of NLP lie in a number of disciplines, viz. computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, etc. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, user interfaces,multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems and so on..

****

****

The Anatomy of a Tweet

*import json*

*with open('mytweets.json', 'r') as f:*

*line = f.readline() # read only the first tweet/line*

*tweet = json.loads(line) # load it as Python dict*

*print(json.dumps(tweet, indent=4)) # pretty-print*

The key attributes are the following:

* text: the text of the tweet itself
* created\_at: the date of creation
* favorite\_count, retweet\_count: the number of favourites and retweets
* favorited, retweeted: boolean stating whether the authenticated user (you) have favourited or retweeted this tweet
* lang: acronym for the language (e.g. “en” for english)
* id: the tweet identifier
* place, coordinates, geo: geo-location information if available
* user: the author’s full profile
* entities: list of entities like URLs, @-mentions, hashtags and symbols
* in\_reply\_to\_user\_id: user identifier if the tweet is a reply to a specific user
* in\_reply\_to\_status\_id: status identifier id the tweet is a reply to a specific status

As you can see there’s a lot of information we can play with. All the \*\_id fields also have a \*\_id\_str counterpart, where the same information is stored as a string rather than a big int (to avoid overflow problems). We can imagine how these data already allow for some interesting analysis: we can check who is most favourited/retweeted, who’s discussing with who, what are the most popular hashtags and so on. Most of the goodness we’re looking for, i.e. the content of a tweet, is anyway embedded in the text, and that’s where we’re starting our analysis.

We start our analysis by breaking the text down into words. Tokenisation is one of the most basic, yet most important, steps in text analysis. The purpose of tokenisation is to split a stream of text into smaller units called tokens, usually words or phrases. While this is a well understood problem with several out-of-the-box solutions from popular libraries, Twitter data pose some challenges because of the nature of the language.

## ***How to Tokenise a Tweet Text***

|  |
| --- |
| *from nltk.tokenize import word\_tokenize*  *tweet = 'RT @marcobonzanini: just an example! :D* [*http://example.com*](http://example.com/) *#NLP'*  *print(word\_tokenize(tweet))*  *# ['RT', '@', 'marcobonzanini', ':', 'just', 'an', 'example', '!', ':', 'D', 'http', ':', '//example.com', '#', 'NLP']* |
|  |

You will notice some peculiarities that are not captured by a general-purpose English tokeniser like the one from NLTK: @-mentions, emoticons, URLs and #hash-tags are not recognised as single tokens. The following code will propose a pre-processing chain that will consider these aspects of the language.

The core component of the tokeniser is the regex\_str variable, which is a list of possible patterns. In particular, we try to capture some emoticons, HTML tags, Twitter @usernames (@-mentions), Twitter #hashtags, URLs, numbers, words with and without dashes and apostrophes, and finally “anything else”. Please take a moment to observe the regexp for capturing numbers: why don’t we just use \d+? The problem here is that numbers can appear in several different ways, e.g. 1000 can also be written as 1,000 or 1,000.00 — and we can get into more complications in a multi-lingual environment where commas and dots are inverted: “one thousand” can be written as 1.000 or 1.000,00 in many non-anglophone countries. The task of identifying numeric tokens correctly just gives you a glimpse of how difficult tokenisation can be.

The regular expressions are compiled with the flags re.VERBOSE, to allow spaces in the regexp to be ignored (see the multi-line emoticons regexp), and re.IGNORECASE to catch both upper and lowercases. The tokenize() function simply catches all the tokens in a string and returns them as a list. This function is used within preprocess(), which is used as a pre-processing chain: in this case we simply add a lowercasing feature for all the tokens that are not emoticons (e.g. :D doesn’t become :d)[1][[4].

**Chapter 4:Data visualization**

Data visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software.

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data you see and how it’s processed.

***Data visualization: A wise investment in your big data future***

With big data there’s potential for great opportunity, but many retail banks are challenged when it comes to finding value in their big data investment. For example, how can they use big data to improve customer relationships? How – and to what extent – should they invest in big data?

In this Q&A with Simon Samuel, Head of Customer Value Modeling for a large bank in the UK, we examine these and other big data issues that confront retail bankers.

### *Why is data visualization important?*

Because of the way the human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than poring over spreadsheets or reports. Data visualization is a quick, easy way to convey concepts in a universal manner – and you can experiment with different scenarios by making slight adjustments[9][10].

Data visualization can also:

* Identify areas that need attention or improvement.
* Clarify which factors influence customer behavior.
* Help you understand which products to place where.
* Predict sales volumes.

***Matplotlib :***

[matplotlib.pyplot](https://matplotlib.org/api/pyplot_api.html#module-matplotlib.pyplot) is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g. creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In [matplotlib.pyplot](https://matplotlib.org/api/pyplot_api.html" \l "module-matplotlib.pyplot" \o "matplotlib.pyplot) various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that “axes” here and in most places in the documentation refers to the *axes* [part of a figure](http://matplotlib.org/faq/usage_faq.html#parts-of-a-figure) and not the strict mathematical term for more than one axis)[11].

use matplotlib to plot the chart

*plt.pie(*

*x=sizes,*

*shadow=True,*

*colors=colors,*

*labels=labels,*

*startangle=90*

*)*

*plt.title("Sentiment of {} Tweets about {}".format(number, query))*

*plt.show()*

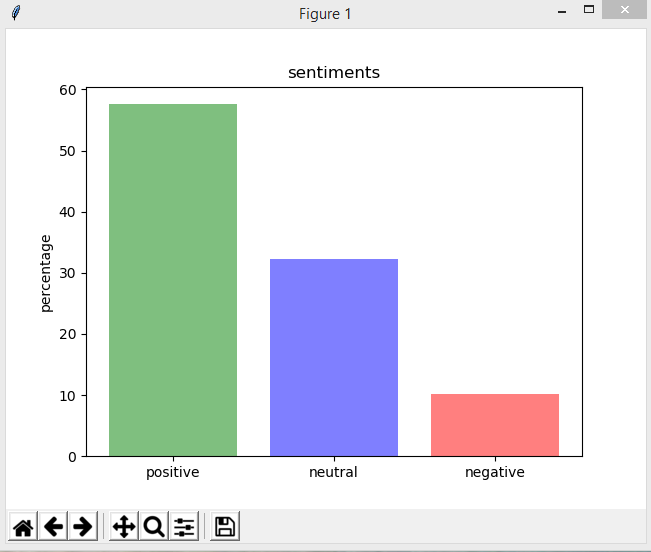
***NumPy*** *:*

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

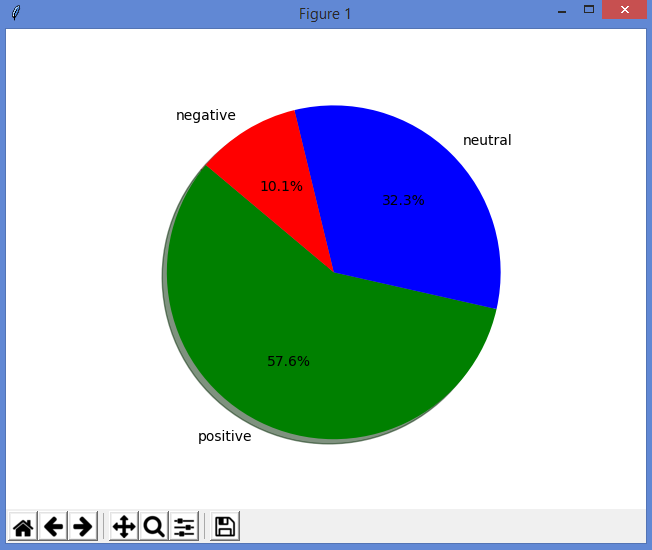
* a powerful N-dimensional array object
* sophisticated (broadcasting) functions
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

***Bar-Graph:***



***Pie-Chart:***

****

**Chapter 5:Naive Bayes algorithm**

***What is Naive Bayes algorithm?***

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



Above,

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).

P(c) is the prior probability of class.

P(x|c) is the likelihood which is the probability of predictor given class.

P(x) is the prior probability of predictor.

***Pros:***

It is easy and fast to predict class of test data set. It also perform well in multi class prediction

When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.

It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

***Cons:***

If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.

Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

4 Applications of Naive Bayes Algorithms

Real time Prediction: Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.

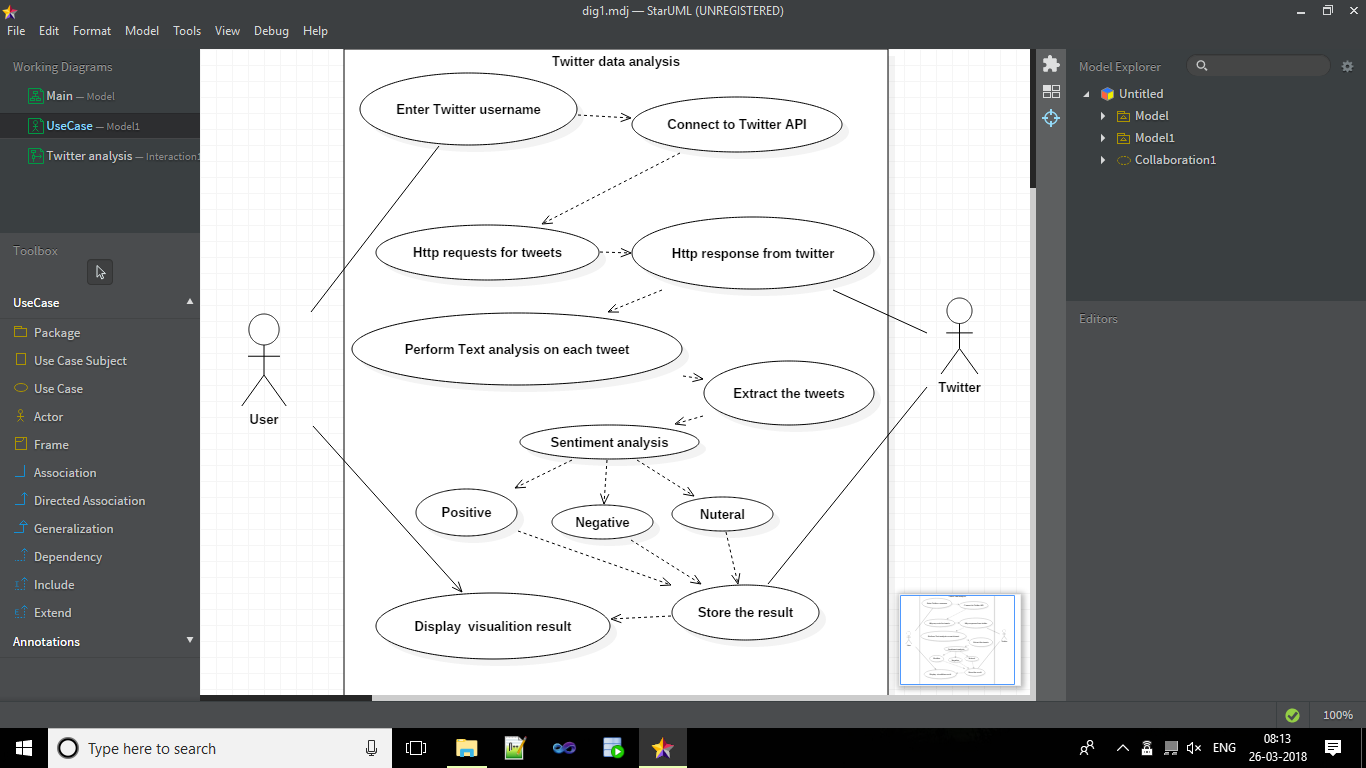
Multi class Prediction: This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.

Text classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

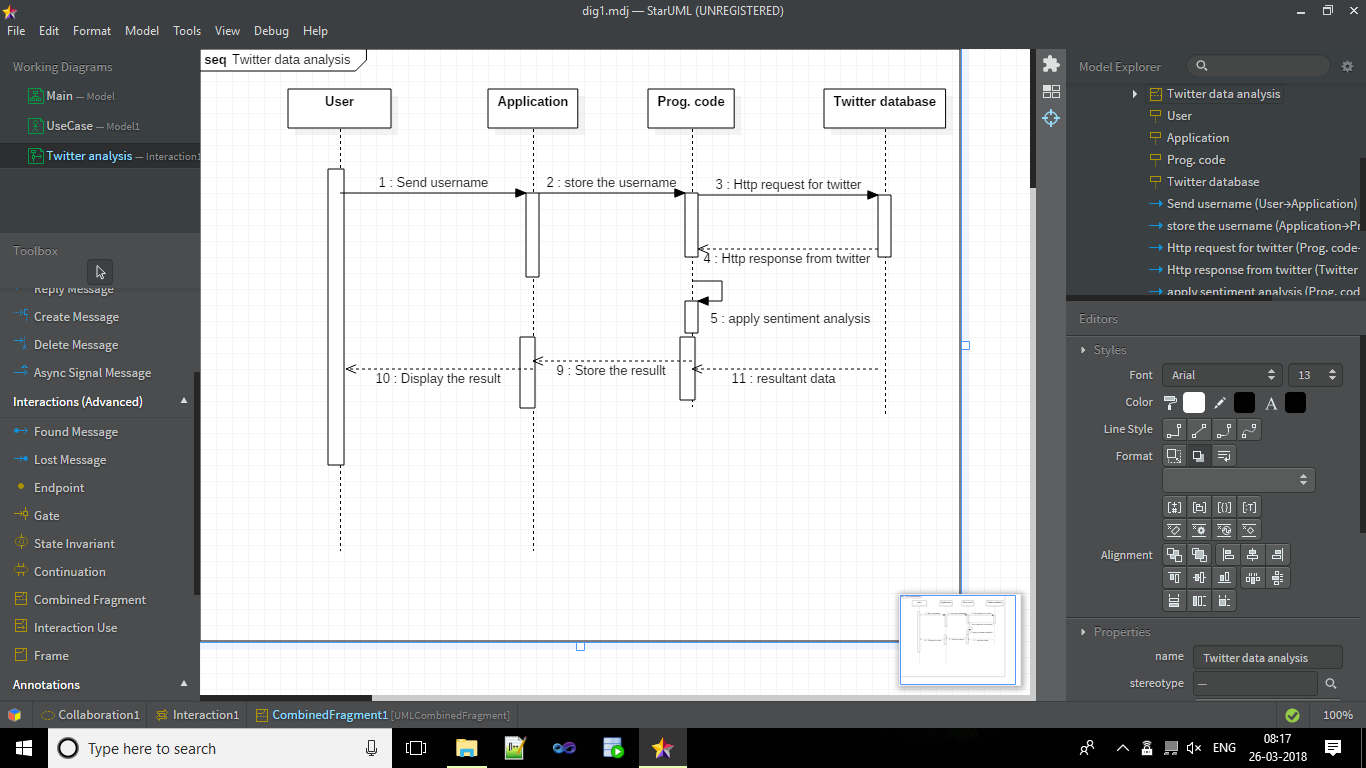
Recommendation System: Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not[8].

**List of Diagrams**

***Use-Case Diagram:***

******

***Sequence Diagram:***

******

**Conclusion :**

In this project we have worked on sentiment analysis on the twitter data using python language and using machine learning algorithms. In Sentiment Analysis we have separated the different types of tweets like positive, negative and neutral by using artificial intelligence and visualized them using python GUI in proper manner.

Right now we have worked with only the very simplest unigram models. The task of sentiment analysis, especially in the domain of micro-bloging, is still in the developing stage and far from complete. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance.

**References:**

1. https://textblob.readthedocs.io/en/dev/api\_reference.html#textblob.blob.TextBlob.sentimet
2. https://dev.to/rodolfoferro/sentiment-analysis-on-trumps...
3. https://www.analyticsvidhya.com/blog/2017/01/beginners-guide-to-create-beauti...
4. https://marcobonzanini.com/2015/05/17/mining-twitter-data-with-python-part-6-sentiment- analysis-basics/
5. <https://apps.twitter.com/>
6. <https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/>
7. <https://en.wikipedia.org/wiki/JSON>
8. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
9. https://en.wikipedia.org/wiki/Data\_visualization
10. https://www.scnsoft.com/blog/big-data-visualization-techniques
11. https://matplotlib.org/users/pyplot\_tutorial.html