**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR nagar, MSRIT post, Bangalore-54

A Dissertation Report on

Twitter Sentiment Analysis

Submitted by

Lakshmishree.C 1MS12CS050

Neha.P 1MS12CS143

Niharika.W.M 1MS12CS065

*in partial fulfillment for the award of the degree of*

# *Bachelor of Engineering in Computer Science & Engineering*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**M.S.RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu), **May 2015**

# ABSTRACT

* Our day-today life has always been influenced by what people think. The explosion of Web 2.0 has led to increased activity in blogging, social networking, etc.
* Sentiment analysis also known as opinion analysis is the task of finding the opinions and affinity of people towards specific topics of interest. Be it a product or a movie, opinions of people matter, and it affects the decision-making process of people. The first thing a person does when he or she wants to buy a product online is to see the kind of reviews and opinions that people have written. The sentiment of the tweets of a particular subject has multiple usages, including stock market analysis of a company, movie reviews, in psychology to analyze the mood of people that has a variety of applications, and so on
* The classification of tweets is done by employing the Naïve Bayes Algorithm. Since this is a supervised machine learning algorithm, a publicly available dataset of twitter messages is used to train the Naïve Bayes classifier. The pre-processing of the dataset done to achieve high accuracy is also described. The primary focus of this project is the usage of a novel feature vector of unigrams to train the machine learning classifier and test it.
* The two types of sentiments considered in this classification experiment are positive and negative sentiments.

#### Contents

***Acknowledgements i***

***Abstract ii***

***List of Figures* 6**

1 **INTRODUCTION 7**

* 1. General Introduction
  2. Statement of the Problem
  3. Objectives of the project
  4. Project deliverables
  5. Current Scope
  6. Future Scope

1. **PROJECT ORGANIZATION 10**
   1. Software Process Models
2. **LITERATURE SURVEY 11**

3.1 Introduction

3.2 Main Body

3.3 Conclusion of Survey

1. **SOFTWARE REQUIREMENT SPECIFICATIONS 14**

4.1 Product Overview

4.2 External Interface Requirements

4.2.1 User Interfaces

4.2.2 Hardware Interfaces

4.2.3 Software Interfaces

4.2.4 Communication Interfaces

4.3 Functional Requirements

4.4 Software System Attributes

4.4.1 Reliability

4.4.2 Availability

4.4.3 Security

4.4.4 Portability

4.4.5 Maintainability

4.4.6 Performance

4.5 Performance Requirements

4.6 Database Requirement

4.7 Design Constraints

4.8 Other Requirements

1. **DESIGN 18**
   1. Introduction
   2. Architecture Design
   3. Class Diagram and Classes (represent Inheritance, Aggregation and Association)
   4. Sequence Diagram
   5. Data flow diagram
2. **IMPLEMENTATION 23**
   1. Tools Introduction
   2. Technology Introduction
   3. Overall view of the project in terms of implementation
   4. Explanation of Algorithm and how it is been implemented
   5. Information about the implementation of Modules
3. **TESTING 33**

**7.1** Results and Snapshots

1. **CONCLUSION & SCOPE FOR FUTURE WORK 36**
2. **REFERENCES 38**

**ACKNOWLEDGEMENT**

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We would like to express our sincere gratitude to our Head Of Department, Dr. K.G.Srinivasa , whose contribution in stimulating suggestions and encouragement,  helped us to successfully complete our project.

**LIST OF FIGURES**

1. ITERATIVE PROCESS MODEL
2. SYSTEM ARCHITECTURE
3. CLASS DIAGRAM
4. DATAFLOW DIAGRAM
5. SEQUENCE DIAGRAM
6. STEPS FOLLOWED IN TWITTER SENTIMENT ANALYSIS
7. **INTRODUCTION**
   1. **GENERAL INTRODUCTION**

Sentiment Analysis is a Natural Language Processing and Information Extraction task that aims to obtain writer’s feelings expressed in positive or negative comments, questions and requests, by analyzing a large numbers of documents. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall tonality of a document. In recent years, the exponential increase in the Internet usage and exchange of public opinion is the driving force behind Sentiment Analysis today. The Web is a huge repository of structured and unstructured data. The analysis of this data to extract latent public opinion and sentiment is a challenging task.

**The Consumer’s Perspective**

While taking a decision it is very important for us to know the opinion of the people around us. Earlier this group used to be small, with a few trusted friends and family members. But, now with the advent of Internet we see people expressing their opinions in blogs and forums. These are now actively read by people who seek an opinion about a particular entity (product, movie etc.). Thus, there is a plethora of opinions available on the Internet. From a consumers’ point of view extracting opinions about a particular entity is very important. Trying to go through such a vast amount of information to understand the general opinion is impossible for users just by the sheer volume of this data. Hence, the need of a system that differentiates between good reviews and bad reviews. Further, labeling these documents with their sentiment would provide a succinct summary to the readers about the general opinion regarding an entity.

**The Producer’s Perspective**

With the explosion of Web 2.0 platforms such as blogs, discussion forums, etc., consumers have at their disposal, a platform to share their brand experiences and opinions, positive or negative regarding any product or service. According to Pang and Lee (2008) these consumer voices can wield enormous influence in shaping the opinions of other consumers and, ultimately, their brand loyalties, their purchase decisions, and their own brand advocacy. Since the consumers have started using the power of the Internet to expand their horizons, there has been a surge of review sites and blogs, where users can perceive a product’s or service’s advantages and faults. These opinions thus shape the future of the product or the service. The vendors need a system that can identify trends in customer reviews and use them to improve their product or service and also identify the requirements of the future.

**The Societies’ Perspective**

Recently, certain events, which affected Government, have been triggered using the Internet. The social networks are being used to bring together people so as to organize mass gatherings and oppose oppression. On the darker side, the social networks are being used to insinuate people against an ethnic group or class of people, which has resulted in a serious loss of life. Thus, there is a need for Sentiment Analysis systems that can identify such phenomena and curtail them if needed.

* 1. **STATEMENT OF THE PROBLEM**

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.

* 1. **OBJECTIVES**

The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream. For a given tweet we classify whether the tweet is of positive, negative, or neutral sentiment. The project heavily relies 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 unlabeled data samples (tweets).

* 1. **PROJECT DELIVERABLES**

The project deliverables include:

1. A database of 1 lakh tweets obtained for hashtags involving smartphones.
2. Sentiment Analysis of each tweet in the database.
3. A graphical visualization depicting the most preferred smartphone.
   1. **CURRENT SCOPE**

The twitter sentiment analyzer designed by us can predict the sentiment of the tweets and also display a graph conveying the most preferred smartphone.

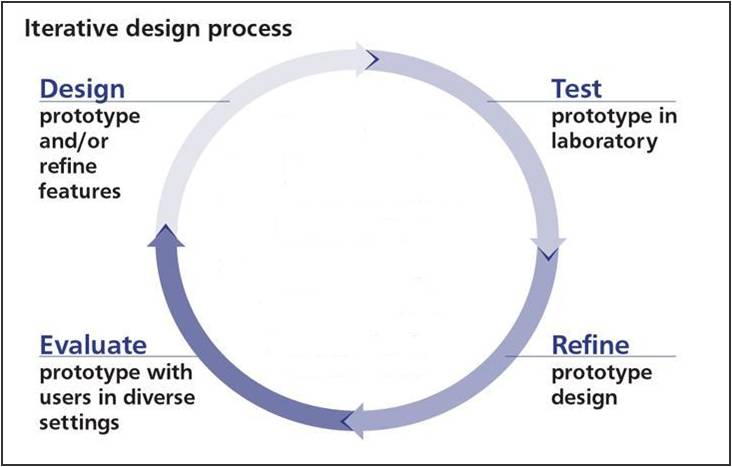
**1.6 FUTURE SCOPE**

Machine learning techniques perform well for classifying sentiment in tweets. The accuracy of the system could be still improved, by using a bigger dataset. This project focuses on tweets in the English language. The same approach can be used to classify sentiment in other languages also.

1. **PROJECT ORGANIZATION**
   1. **SOFTWARE PROCESS MODELS**

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).

Typically iterative development is used in conjunction with incremental development in which a longer [software development cycle](http://searchsoftwarequality.techtarget.com/definition/systems-development-life-cycle) is split into smaller segments that build upon each other.



**FIGURE 1: ITERATIVE PROCESS MODEL**

1. **LITERATURE SURVEY**
   1. **INTRODUCTION**

Sentiment analysis of conventional text is a relatively old problem. However, recent years have witnessed increased interests in sentiment analysis of microblogs. This chapter describes the previous work in sentiment analysis. In particular, we will focus more on recent work of sentiment analysis of microblogs, identify their problems, and provide a cross comparison at the end.

* 1. **MAIN BODY**

Twitter is a micro-blogging website and ranks second amongst the present social media websites (Prelovac 2010). A micro-blog allows users to exchange small elements of content such as short sentences, individual pages, or video links. Alec et al. (2009) provide one of the first studies on sentiment analysis on micro-blogging websites. Barbosa et al. (2010) and Bermingham et al. (2010) both cite noisy data as one of the biggest hurdles in analyzing text in such media. Alec et al. (2009) describes a distant supervision-based approach for sentiment classification. They use hashtags in tweets to create training data and implement a multi-class classifier with topic-dependent clusters. Barbosa et al. (2010) propose an approach to 38 sentiment analysis in Twitter using POS-tagged n-gram features and some Twitter specific features like hashtags. Our system is inspired from C-Feel-IT, a Twitter based sentiment analysis system (Joshi et al., 2011). General spam filtering techniques include approaches that implement Bayesian filter (Sahami, 1998; Graham, 2006), or SVM-based filters along with various boosting algorithms to further enhance the accuracies (Drucker et al., 1999). Twitter is a very noisy medium. However, not much work has been done in the area of text normalization in the social media especially pertaining to Twitter. But there has been some work in the related area of SMS-es. Aw et al., (2006) and Raghunathan et al., (2009) used a MT-based system for text normalization. Choudhury et al., (2007) deployed a HMM for word-level decoding in SMS-es; while Catherine et al., (2008) implemented a combination of both by using two normalization systems: first a SMT model, and then a second model for speech recognition system. Another approach to text normalization has been to consider each word as a corrupt word after being passed through a noisy channel, which essentially boils down to spell-checking itself. Mayes (1991) provide one such approach. Church et al., (1991) provide a more sophisticated approach by associating weights to the probable edits required to correct the word. We follow the approach of Church et al., (1991) and attempt to infuse linguistic rules within the minimum edit distance (Levenstein, 1966). We adopt this simpler approach due to the lack of publicly available parallel corpora for text normalization in Twitter. Unlike in Twitter, there has been quite a few works on general entity specific sentiment analysis. Nasukawa et al., (2003) developed a lexicon and sentiment transfer rules to extract sentiment phrases. Mullen et al., (2004) used Osgood and Turney values to extract value phrases, i.e. sentiment bearing phrases from the sentence. Many approaches have also tried to leverage dependency parsing in entity-specific SA. Mosha (2010) uses dependency parsing and shallow semantic analysis for Chinese opinion related expression extraction. Wu et al., (2009) used phrase dependency parsing for opinion mining. Mukherjee et al. (2012) exploit dependency parsing for graph based clustering of opinion expressions about various features to extract the opinion expression about a target feature. We follow a dependency 39 parsing based approach for entity specific SA as it captures long distance relations, syntactic discontinuity and variable word order, as is prevalent in Twitter. The works (Alec et al., 2009; Read et al., 2005; Pak et al., 2010; Gonzalez et al. (2011)) evaluate their system on a dataset crawled and auto-annotated based on emoticons, hashtags. We show, in this work, that a good performance on such a dataset does not ensure a similar performance in a general setting.

* 1. **CONCLUSION**

This report discusses in details the various approaches to Sentiment Analysis, mainly Machine Learning. We have seen the applications of machine learning techniques like Naïve Bayes, Maximum Entropy, Support Vector Machines. As all of these are bag-of-words model, they do not capture context and do not analyze the discourse which is absolutely essential for SA. Thus machine learning models with a proper kernel that can capture the context will play an important role in Sentiment Analysis. Feature engineering, as in several Machine Learning and Natural Language Processing applications, plays a vital role in Sentiment Analysis. We have seen the use of phrases as well as words as features. It has been seen that Adjectives as word features can capture majority of the sentiment. Use of topic oriented features and Value Phrases play a significant role to detect sentiment when the domain of application is known. It is also seen that use of lemmas capture sentiment better than using unigrams.

**4. SOFTWARE REQUIREMENT SPECIFICATION**

* 1. **EXTERNAL INTERFACE REQUIREMENTS**

**4.1.1 User Interfaces**

It will be simple and easy to understand. Controls which allow the user to interact with the application will be clear and imply their functionality within the application.

The interface, a graph will provide a visual representation of the output. This graphic will consist of a simple gauge which shows the current mood of the Twitter community on a given topic. This will be done by displaying the percentage of the Twitter users who are currently for or against the topic being analyzed. It will also display the total number of Tweets which have been processed in order to calculate this output

**4.1.2 Hardware Interfaces**

The software is intended to be a standalone, single-user system. The hardware devices used are: SQLite database server with intensive use of memory space and Windows users’ computers.

* + 1. **Software Interfaces**

The software requirements have to be compatible with Windows operating system. For tweet extraction, twitter is the only source and uses Streaming API that offers high throughput. Using this API is appropriate as it enables to retrieve real time information. Additionally, this continuous stream of tweets can be extracted without missing any information. SQLite database is needed for saving of extracted tweets. Anaconda with appropriate libraries is used to edit and write python programs. Spyder is used which an open source cross-platform IDE for programming in the Python.

**Inputs** - The software will receive input from the Twitter API i.e. the tweets extracted.

**Outputs** - The output will portray the current opinion of the Twitter community on a given topic in the form of a simple graph.

**Operating System** - The software will run on the Windows operating system.

**4.1.4 Communication Interfaces**

Internet connection and a web browser are required in order to extract tweets relating to a particular hashtag.

* 1. **FUNCTIONAL REQUIREMENTS**
     1. **Retrieving input**

The software will receive Tweets as input. Tweets will be retrieved with the Twitter Streaming API.

* + 1. **Sentiment Analysis**

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

* + 1. **Output**

The software must output real time data in the form of a simple graph. In addition, the software may output a graph of mood trends over time, as well as additional statistics pertaining to a topic (average sentiment over all analysis sessions and total number of tweets processed). This output should be clear and easy to understand.

* 1. **SOFTWARE SYSTEM ATTRIBUTES**
     1. **Reliability**

The software will meet all of the functional requirements without any unexpected behavior. At no time should the graphical output display incorrect without alerting the user to potential errors. It should be tested and debugged completely. All exceptions should be well handled.

* + 1. **Availability**

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

* + 1. **Security**

The software should never disclose any personal information of Twitter users, and should collect no personal information from its own users. Also, only authorized persons should be allowed to access the database of tweets. Security can be ensured by providing authentication.

* + 1. **Portability**

This software will work on Windows 8 or higher, so it is forward compatible with Windows OSs**.**

* + 1. **Maintainability**

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

* + 1. **Performance**

This software has to be implemented on Windows for best performance. The time taken for the software to predict the response based on tweets extracted may increase as the size of tweets increases. There can exist only one user terminal to view the result of the algorithm.

* 1. **PERFORMANCE REQUIREMENTS**

**Response time**

As size of tweet collection increases, the time taken for the program to determine the opinion from the tweets is also increased**.**

**System Resource Consumption**

Resource consumption of this application should not reach an amount that renders the computer device unusable. The application should be capable of operating in the background should the user wish to utilize other applications.

* 1. **DATABASE REQUIREMENT**

A database that is capable of holding a large number of extracted tweets – up to 1 lakh tweets. It should allow efficient add, insert, update and delete operations, ensure security, avoid redundancy and have a reasonable access time.

* 1. **DESIGN REQUIREMENT**

**Processing Power**

Requires high speed machine for data capturing from various sources, classifying the sentiment polarity of large data and extracting topics.

**Deployment Point**

Built to be deployed at a single user host system.

**Operating Platform**

May work for several distributions of Linux and Windows PCs.

**4.8 OTHER REQUIREMENTS**

**Safety Requirements**

For the safety requirements nothing but an operation of weekly backups for the data base should take place.

**5. DESIGN**

**5.1 PRODUCT OVERVIEW**

Our twitter sentiment analyzer analyses the tweets and predicts the sentiment of the tweets to be positive, negative or neutral. We have used SAP Lumira to interpret the results, which is in a graphical form. A graphical representation is more easily understandable that a textual report.

**5.2 ARCHITECTURE DESIGN**

1. The system is structured in such a way that each step is processed in sequence. In order to complete this project the core requirement is a Twitter account, without a Twitter account one is unable to access the Twitter API and to request data from the Twitter stream. The requirement for creating a Twitter account are the user’s full name (first and last), a valid email address, a desired password and to agree to the Terms of Service for Twitter.

The next step is to create a new application. Select the “Create New App” button on the Twitters Apps page and follow the instructions on screen. The name of the created application must be unique; it cannot have been 14 take taken already by another user. If you try to create an application with a preexisting name an error will occur and the page will prompt you for a new name.

Once this application is created you will have to access the API keys tab to decide on the level of access required: ‘read only’, ‘read and write’, or ‘read, write and access direct messages’ and to create the access token. Generating the access token will give you the codes required to safely and securely access the Twitter stream. The extracted tweets are stored in a SQLite database.

1. Pre-processing

Preprocessing eliminates the part which does not contribute significantly to the polarity detection. There are many nooks and crooks of the social media datasets also known as tweets for Twitter. Tweets often contain usernames of account holder (@nirajp) which are replaced with the generic token USERNAME. Links(http://goo.gl/nirajp) are eliminated or replaced with the generic token URL. Additionally, suggested further more preprocessing of tweets to reduce the feature which includes converting tweets to lower case characters to remove unevenness.’#’ symbol used to denote hash tags are eliminated while keeping the succeeding hash tag word. Stop words such as a, is, which do not contribute significantly to polarity detection are eliminated. Punctuation marks and additional white spaces are also eliminated. Two or more repetitive letters in a word are eliminated.e.g.: Happy is represented as haaappy or haaaaaaappy to stress emotion on social media platform is converted to ’happy’. Care is also taken that words must start with an alphabet. For the sake of simplicity, all those words which don’t start with an alphabet are removed to reduce feature e.g. 21st, 7:30 pm

1. Feature Engineering

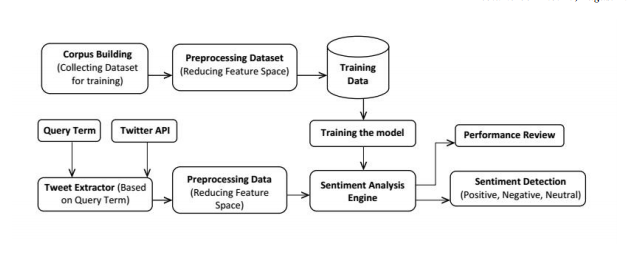
Feature Extraction is an extremely basic and essential task for Sentiment Analysis. Converting a piece of text to a feature vector is the basic step in any data driven approach to Sentiment Analysis Unigram For text classification purpose, the unigram model was used which selects individual words from the data.Apple is opening up the iPhone SDK. I’m stoked!, for instance, contains following unigrams: Apple, opening, iPhone, SDK, stoked ,etc

1. Training

For training purpose, the polarity labelled data from corpus is first parsed and relevant features are extracted from it to build the feature vector.This vector is used to create a Feature List which is a list of all the features of all the data items in dataset used for training, this list is stored in a text file on secondary memory for further use in both the training and classification

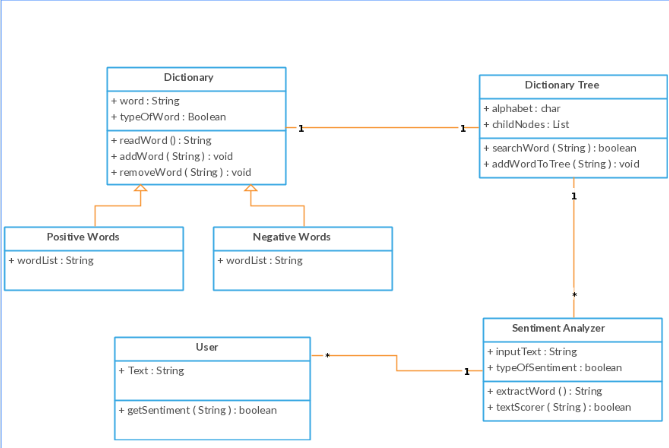
1. Naive Bayesian Classifier.

Naive Bayesian Text Classification algorithm is used for the purpose of classification of given trained model. It is the probabilistic approach to the text classification. Here the class labels are known and the goal is to create probabilistic models, which can be used to classify new texts. It is specifically formulated for text and makes use of text specific characteristics. The Naive Bayesian classifier treats each document as a ”bag of words” and the generative model makes the following assumptions: firstly, words of a document are generated independently of context, and, secondly, the probability of the word is independent of its position. This is why the name naive was used for this algorithm. In real text documents the words often correlate with each other and the position of the word in text may play role.

****

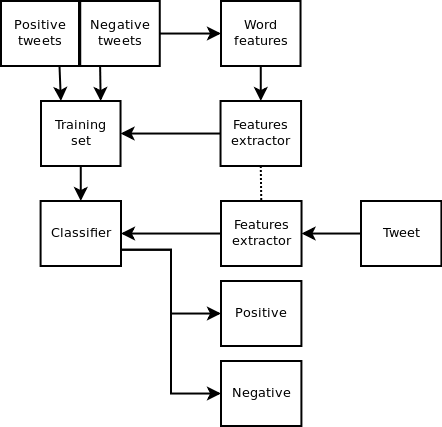
**FIGURE 2: SYSTEM ARCHITECTURE**

**5.3 CLASS DIAGRAM**

****

**FIGURE 3: CLASS DIAGRAM**

**5.4 DATAFLOW DIAGRAM**

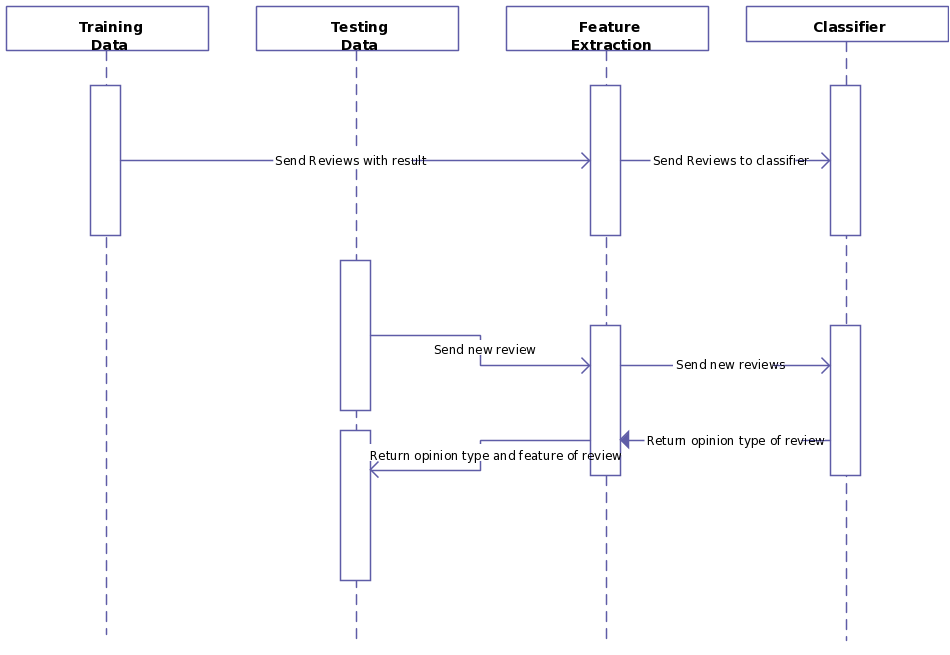


**FIGURE 4: DATAFLOW DIAGRAM**

**5.5 SEQUENCE DIAGRAM**

The following are the processes used in the sequence diagram to show the interaction between them:

1. Training data- this phase involves training set which is manually classified.
2. Testing data- consists of the sampled data from initial data extracted for particular keyword.
3. Feature extraction- input is the tweets from either training set or testing set from which labels are extracted after preprocessing and filtering.
4. Classifier – the feature list is fed to classifier in training stage to train the classifier. The feature list from test set is sent to classifier which gets classified and returned with appended opinion type.(positive, negative or neutral)

****

**FIGURE 5: SEQUENCE DIAGRAM**

**6. IMPLEMENTATION**

**6.1 TOOLS INTRODUCTION**

1. Anaconda is a [freemium](https://en.wikipedia.org/wiki/Freemium) distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Its [package management system](https://en.wikipedia.org/wiki/Package_manager) is *conda*.

1. Spyder (formerly Pydee) is an [open source](https://en.wikipedia.org/wiki/Open-source_software) cross-platform [IDE](https://en.wikipedia.org/wiki/Integrated_development_environment) for scientific programming in the [Python language](https://en.wikipedia.org/wiki/Python_(programming_language)). Spyder integrates [NumPy](https://en.wikipedia.org/wiki/NumPy), [SciPy](https://en.wikipedia.org/wiki/SciPy" \o "SciPy), [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib" \o "Matplotlib) and [IPython](https://en.wikipedia.org/wiki/IPython" \o "IPython), as well as other open source software. In comparison with other IDEs for scientific development Spyder has a unique set of features - cross-platform, open-source, written in Python and available under non-copyleft license. Spyder is extensible with plugins, includes support for interactive tools for data inspection. It is available cross-platform through [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)), on Windows with WinPython.
2. Python 2.7(implementation language)

Python is a general-purpose, interpreted high-level programming language whose design philosophy emphasizes code readability. Its syntax is clear and expressive. Python has a large and comprehensive standard library and more than 25 thousand extension modules.

1. NLTK (language processing modules and validation)

The Natural Language Processing Toolkit (NLTK) is an open source language processing module of human language in python. Created in 2001 as a part of computational linguistics course in the Department of Computer and Information Science atthe University of Pennslyvania. NLTK provides inbuilt support for easy-to-use interfaces over 50 lexicon corpora. NLTK was designed with four goals in mind:

1. Simplicity

Provide and intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annoted language data.

1. Consistency

Provide a uniform framework with consistent interfaces and data structures, and easily guessable method names.

1. Extensibility

Provide a structure into which new software modules can easily be accommodated, including alternative implementations and competing approaches on the same task.

1. Modularity

Provide components that can be used independently without needing to understand the rest of the toolkit.

1. SAP Lumira is a data visualization solution that allows us to combine multiple data sources and discover unique insights. The software lets you understand your data by building visualizations using a drag and drop interface. It can spot trends and identify patterns immediately.

**6.2 TECHNOLOGY INTRODUCTION**

* Data mining is an interdisciplinary subfield of [computer science](https://en.wikipedia.org/wiki/Computer_science). It is the computational process of discovering patterns in large [data sets](https://en.wikipedia.org/wiki/Data_set) ("[big data](https://en.wikipedia.org/wiki/Big_data)") involving methods at the intersection of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), [machine learning](https://en.wikipedia.org/wiki/Machine_learning), [statistics](https://en.wikipedia.org/wiki/Statistics), and [database systems](https://en.wikipedia.org/wiki/Database_system). The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and [data management](https://en.wikipedia.org/wiki/Data_management) aspects, [data pre-processing](https://en.wikipedia.org/wiki/Data_pre-processing), [model](https://en.wikipedia.org/wiki/Statistical_model) and [inference](https://en.wikipedia.org/wiki/Statistical_inference) considerations, interestingness metrics, [complexity](https://en.wikipedia.org/wiki/Computational_complexity_theory) considerations, post-processing of discovered structures, [visualization](https://en.wikipedia.org/wiki/Data_visualization), and [online updating](https://en.wikipedia.org/wiki/Online_algorithm). Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD.
* Machine learning is a subfield of [computer science](https://en.wikipedia.org/wiki/Computer_science) that evolved from the study of [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition) and [computational learning theory](https://en.wikipedia.org/wiki/Computational_learning_theory) in [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). Machine learning explores the study and construction of [algorithms](https://en.wikipedia.org/wiki/Algorithm) that can [learn](https://en.wikipedia.org/wiki/Learning) from and make predictions on [data](https://en.wikipedia.org/wiki/Data). Such algorithms operate by building a [model](https://en.wikipedia.org/wiki/Mathematical_model) from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions.
* Machine learning is closely related to [computational statistics](https://en.wikipedia.org/wiki/Computational_statistics); a discipline that aims at the design of algorithm for implementing statistical methods on computers. It has strong ties to [mathematical optimization](https://en.wikipedia.org/wiki/Mathematical_optimization), which delivers methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit [algorithms](https://en.wikipedia.org/wiki/Algorithm) is infeasible. Example applications include [spam filtering](https://en.wikipedia.org/wiki/Spam_filter), [optical character recognition](https://en.wikipedia.org/wiki/Optical_character_recognition) (OCR), [search engines](https://en.wikipedia.org/wiki/Learning_to_rank) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision). Machine learning is sometimes conflated with [data mining](https://en.wikipedia.org/wiki/Data_mining), although that focuses more on exploratory data analysis. Machine learning and pattern recognition "can be viewed as two facets of the same field."
  1. **OVERALL VIEW OF THE PROJECT IN TERMS OF IMPLEMENTATION**

The following steps are followed in our project:

Data collection, Pre-processing, Training the classifier and Classification











**FIGURE 6: STEPS FOLLOWED IN TWITTER SENTIMENT ANALYSIS**

Data collection: Twitter API

* For classification and training the classifier we need Twitter data. For this purpose we make use of API's twitter provides. Twitter provides two API's; Stream API and REST API.
* The difference between Streaming API and REST APIs are: Streaming API supports long-lived connection and provides data in almost real -time. The REST APIs support short-lived connections and are rate-limited (one can download a certain amount of data [\*150 tweets per hour] but not more per day).
* REST APIs allow access to Twitter data such as status updates and user info regardless of time. However, Twitter does not make data older than a week or so available. Thus REST access is limited to data Twittered not before more than a week. Therefore, while REST API allows access to these accumulated data, Streaming API enables access to data as it is being twittered.
* The search API provides users the ability to access twitter search functionality. It uses GET requests and returns results formatted using ATOM or JSON, JSON is recommended due to compactness.

Preprocessing

The tweets gathered from twitter are a mixture of urls, and other non-sentimental data like hashtags \#", annotation \@" and retweets \RT". To obtain n-gram features, we first have to tokenize the text input. Tweets pose a problem for standard tokenizers designed for formal and regular text. The following figure displays the various intermediate processing feature steps.

Tokenize

For a sample input text say "Today the weather is sunny and beautiful", Tokenizers divide strings into lists of substrings also known as Tokens. Tokenizing the text makes it easy to separate out other unnecessary symbols and punctuations and filter out only those words that can add value to the sentimental polarity score of the text.

Stop words

In information retrieval, it is a common tactic to ignore very common words such as \a", \an", \the", etc. since their appearance in a post does not provide any useful information in classifying a document. Since query term itself should not be used to determine the sentiment of the post with respect to it, every query term is replaced with a QUERY keyword. Although this makes it somewhat of a stop word, it can still be useful when not using a bag-of-words model and the location of the query in relation to other words becomes important.

Training Data

To precisely label the text into their respective classes and thus achieve highest possible accuracy, we plan to train the classifier using pre-labelled twitter data itself.

In our project, we have used a training dataset that was already available on the internet.

**6.4 EXPLANATION OF ALGORITHM AND HOW IT HAS BEEN IMPLEMENTED**

Naive Bayes is a simple model which works well on text categorization Naive Bayes was our first choice.Naive bayes is bayesian probability distribution model based algorithm. In general all bayesian models are derivatives of the well known Bayes Rule, which suggests that the probability of a hypothesis given a certain evidence, i.e. the posterior probability of a hypothesis, can be obtained in tems of the prior probability of the evidence, the prior probability of the hypothesis and the conditional probability of the evidence given the hypothesis.

Mathematically, P(H|E) = P(H)P(E|H) P(E)

where,

P(H|E)- posterior probability of the hypothesis.

P(H)- prior probability of hypothesis.

P(E)- prior probability of evidence.

P(E|H)- conditional probability of evidence of given hypothesis.

Or in a simpler form: Posterior = (P rior) × (Likelihood) Evidence

To explain the concept, let’s take an example. For instance, we have a new tweet to be classified in to one of the positive or negative classes. Given that in the previously classified tweets, positive tweets are twice the number of negative tweets. Since the new tweet’s class is not known, the problem is estimating correctly the class that the tweet is to be categorised in. This can be found out by Bayes rule calculating the probabilities of the likelihood of the tweet to be positive or negative.

We have: P(n|p) = P(n)P(p|n) P(p)

Since there are twice as many positive tweets as negative, it is reasonable to believe that a new case (which hasn’t been observed yet) is twice as likely to have membership positive rather than negative. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of positive tweets and negative tweets, and often used to predict outcomes before they actually happen.

Thus, we can write:

Prior Probability of positive tweetP(p) = No. of positive tweets /Total no. of tweets

Prior Probability of negative tweetP(n) = No. of negative tweets/ Total no. of tweets

Let there be say a total of 6k tweets, 4k of which are positive and 2k negative, our prior probabilities for class membership are(where k = 103 ):

Prior Probability for positive tweet P(p) = 4k/ 6k = 4 /6 = 2/ 3

Prior Probability for negative tweet P(R)= 2k/ 6k = 2/ 6 = 1/ 3

The likelihood of the tweet falling into either of the classes is equal, since we have only two classes. So likelihood of X = 0.5. So now calculating the posterior probability of the new tweet say X, being positive or negative, will be:

• Posterior probability of X being positive = (Prior probability of positive)× (Likelihood of X being positive) = 2 3 × 1 2 = 1 3 = 33.34% chances of X being positive.

• Posterior probability of X being negative = (Prior probability of negative)× (Likelihood of X being negative) = 1 3 × 1 2 = 1 6 = 16.67% chances of X being negative.

Thus this tweet will fall in to the positive class. In our case we would have two hypothesis and many other features on basis of which the one that has the highest probability would be chosen as a class of the tweet whose sentiment is being predicted. After every classification step all the probabilities are again calculated and updated accordingly.

* 1. **INFORMATION ABOUT THE IMPLEMENTATION OF THE MODULES**

1. Process Tweets

## The following is done by the code to preprocess the tweets:

1. Lower Case - Convert the tweets to lower case.
2. URLs - We eliminate all of these URLs via regular expression matching or replace with generic word URL.
3. @username - we can eliminate "@username" via regex matching or replace it with generic word AT\_USER.
4. #hashtag - hash tags can give us some useful information, so it is useful to replace them with the exact same word without the hash. E.g. #nike replaced with 'nike'.
5. Punctuations and additional white spaces - remove punctuation at the start and ending of the tweets. E.g: ' the day is beautiful! ‘replaced with 'the day is beautiful'. It is also helpful to replace multiple whitespaces with a single whitespace

### 

### def processTweet(tweet):

    tweet = tweet.lower()

    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)

    tweet = re.sub('@[^\s]+','AT\_USER',tweet)

tweet = re.sub('[\s]+', ' ', tweet)

tweet = re.sub(r'#([^\s]+)', r'\1', tweet)

tweet = tweet.strip ('\'"')

    return tweet

1. Eliminate Stop Words

Stop words - a, is, the, with etc. The full list of stop words can is stored in another file called stopWordListFileName. These words don't indicate any sentiment and can be removed.

def getStopWordList(stopWordListFileName):

    stopWords = []

    stopWords.append('AT\_USER')

    stopWords.append('URL')

    fp = open(stopWordListFileName, 'r')

    line = fp.readline()

    while line:

        word = line.strip()

        stopWords.append(word)

        line = fp.readline()

    fp.close()

    return stopWords

1. Feature Vector and its extraction

Feature vector is the most important concept in implementing a classifier. A good feature vector directly determines how successful your classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data.

In tweets, we can use the presence/absence of words that appear in tweet as features. In the training data, consisting of positive, negative and neutral tweets, we can split each tweet into words and add each word to the feature vector. Some of the words might not have any say in indicating the sentiment of a tweet and hence we can filter them out. Adding individual (single) words to the feature vector is referred to as 'unigrams' approach.

Some of the other feature vectors also add 'bi-grams' in combination with 'unigrams'. For example, 'not good' (bigram) completely changes the sentiment compared to adding 'not' and 'good' individually. Here, for simplicity, we will only consider the unigrams. Before adding the words to the feature vector, we need to preprocess them in order to filter; otherwise, the feature vector will explode.

def getFeatureVector(tweet, stopWords):

    featureVector = []

    words = tweet.split()

    for w in words:

        #replace two or more with two occurrences

        w = replaceTwoOrMore(w)

        #strip punctuation

        w = w.strip('\'"?,.')

        #check if it consists of only words

        val = re.search(r"^[a-zA-Z][a-zA-Z0-9]\*[a-zA-Z]+[a-zA-Z0-9]\*$", w)

        #ignore if it is a stopWord

        if(w in stopWords or val is None):

            continue

        else:

            featureVector.append(w.lower())

    return featureVector

#start extract\_features

def extract\_features(tweet):

    tweet\_words = set(tweet)

    features = {}

    for word in featureList:

        features['contains(%s)' % word] = (word in tweet\_words)

    return features

1. Generation of Training set

# Generate the training set

training\_set = nltk.classify.util.apply\_features(extract\_features, tweets)

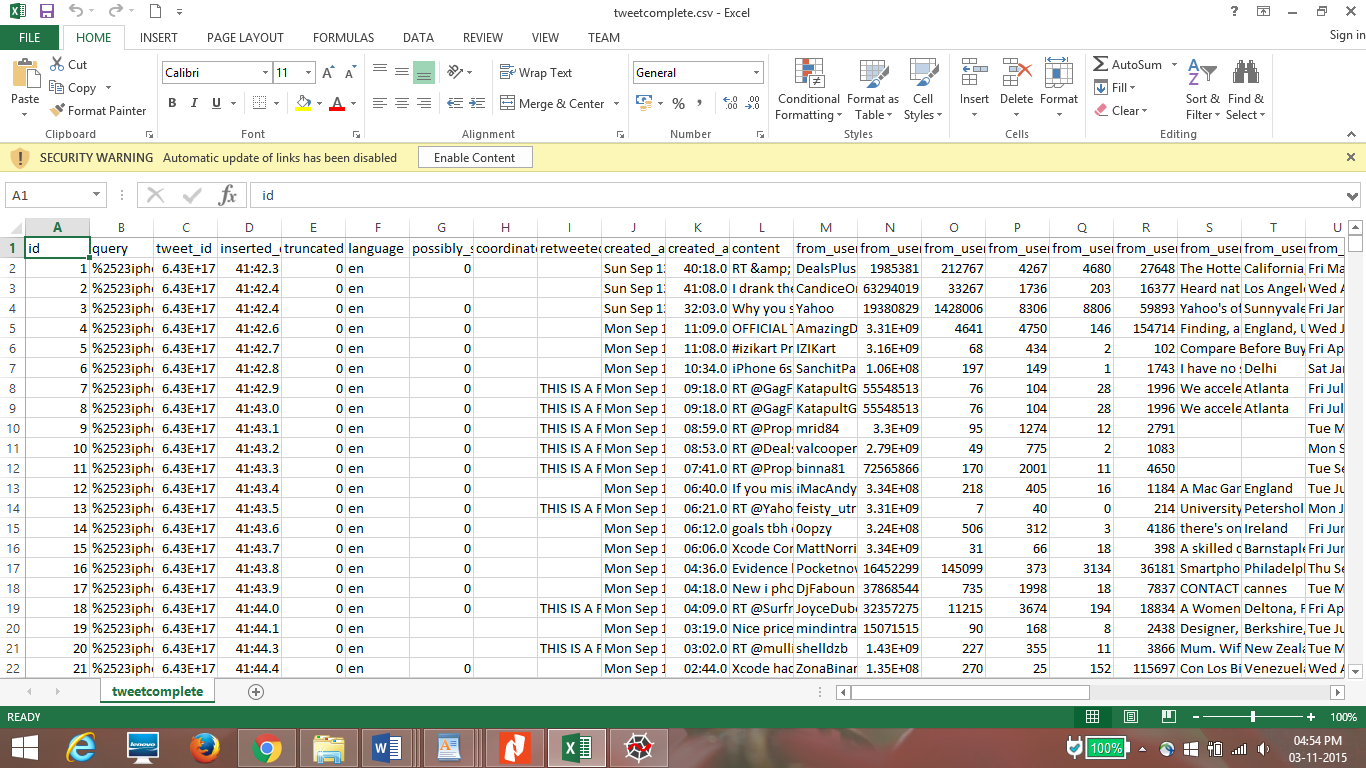
1. Training the Classifier

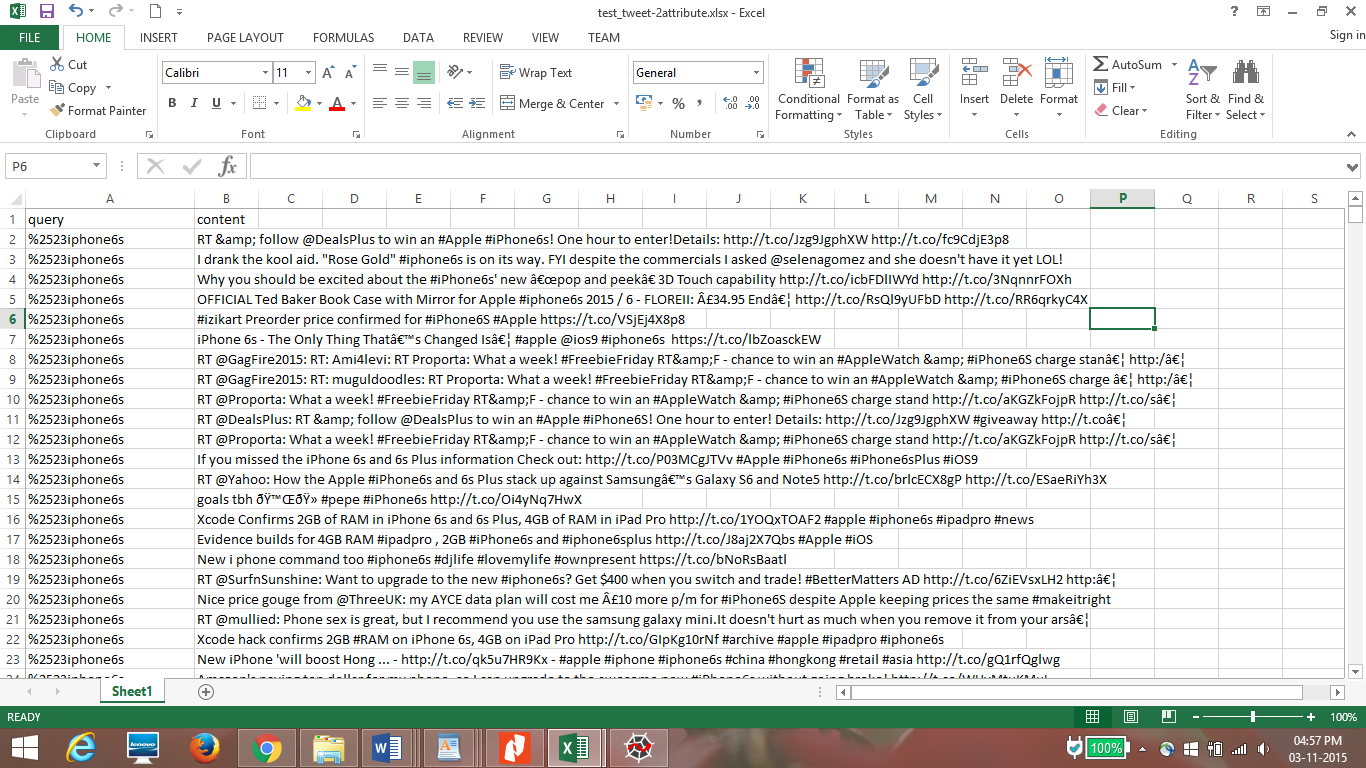
# Train the Naive Bayes classifier

NBClassifier = nltk.NaiveBayesClassifier.train(training\_set)

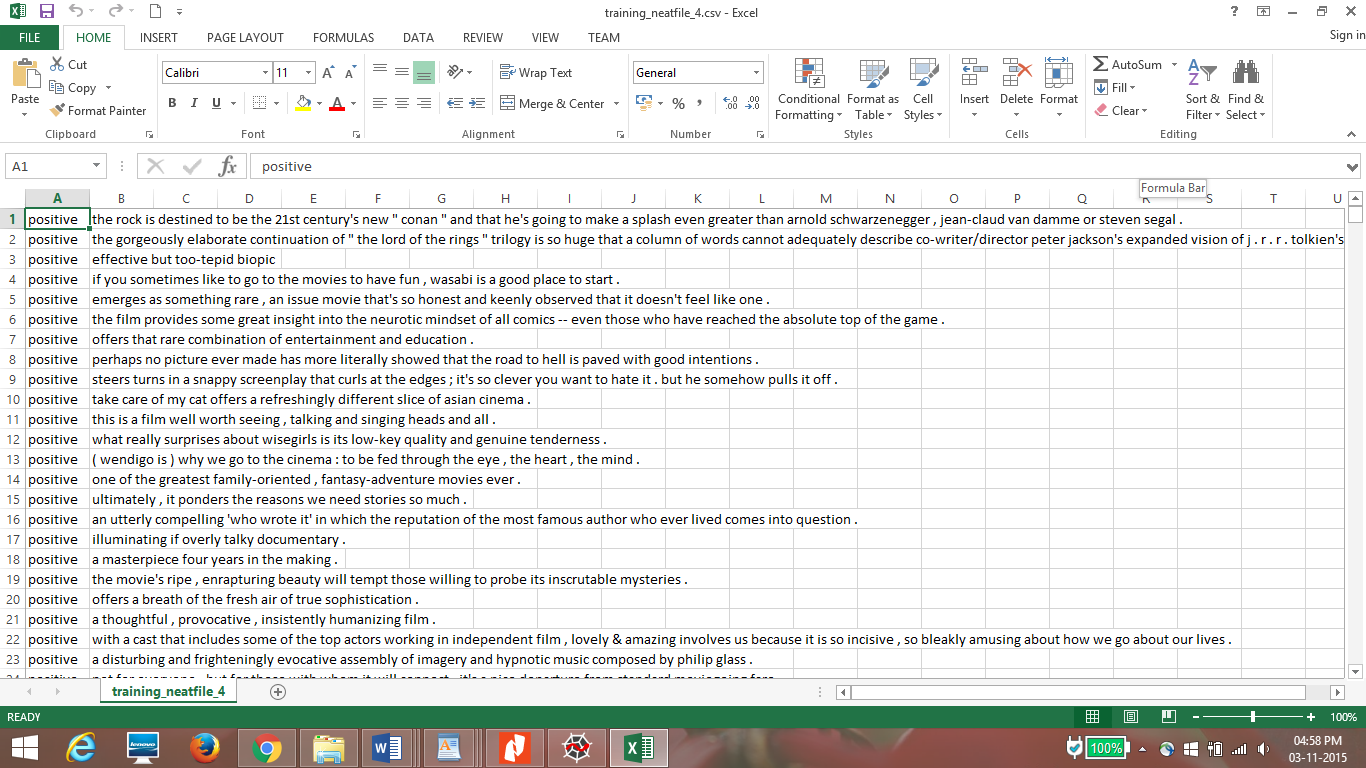
1. **TESTING**
   1. **RESULTS AND SCREENSHOTS**

Collection of tweets related to hashtags such as#iphone6s,#samsunggalaxy,#nexus,etc.

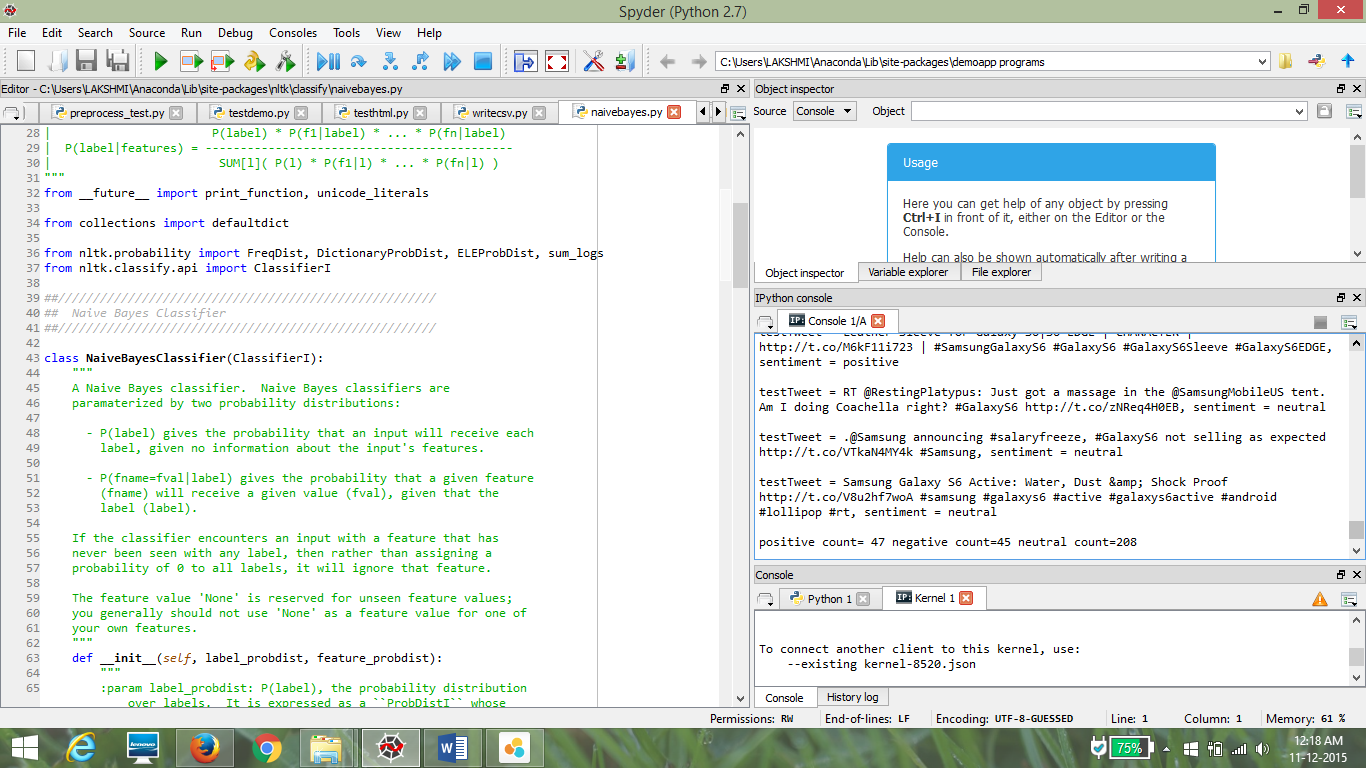




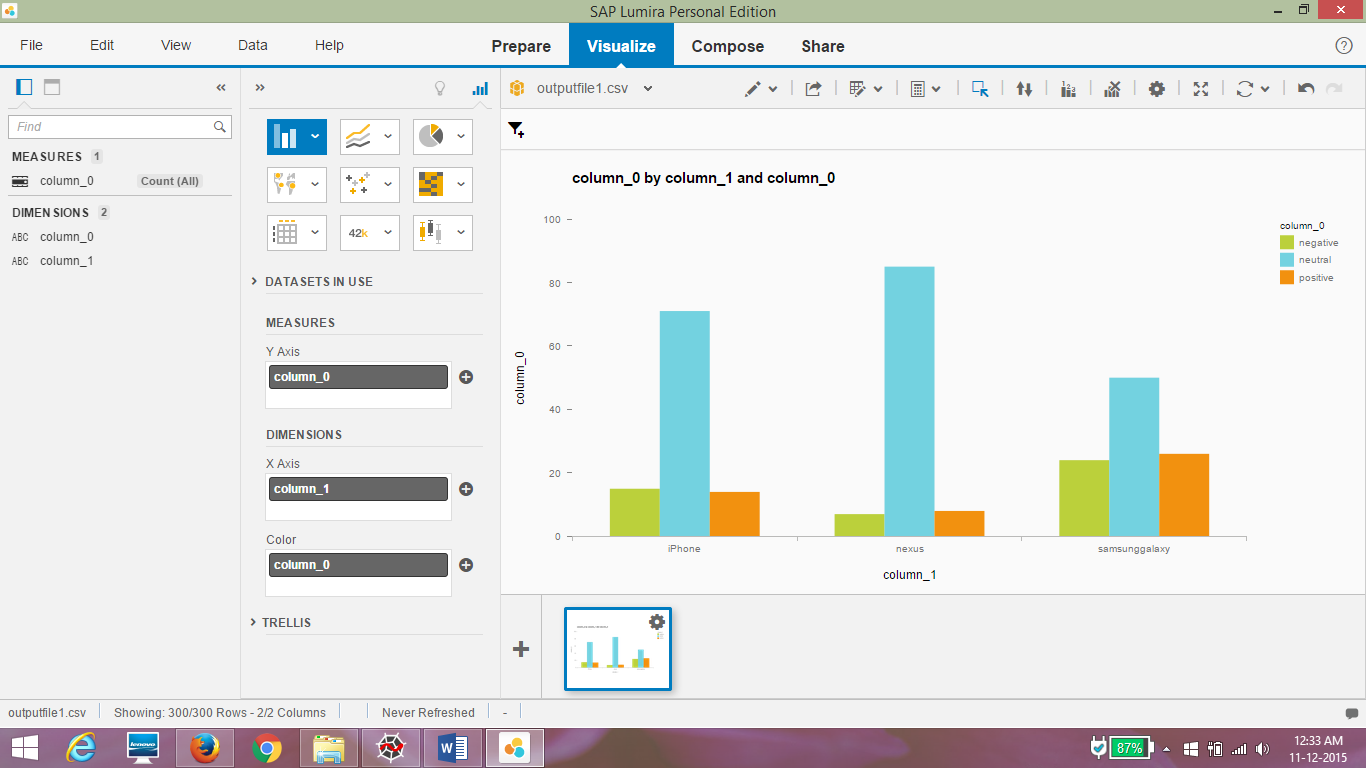
Collection of 20000 training dataset which is manually classified.

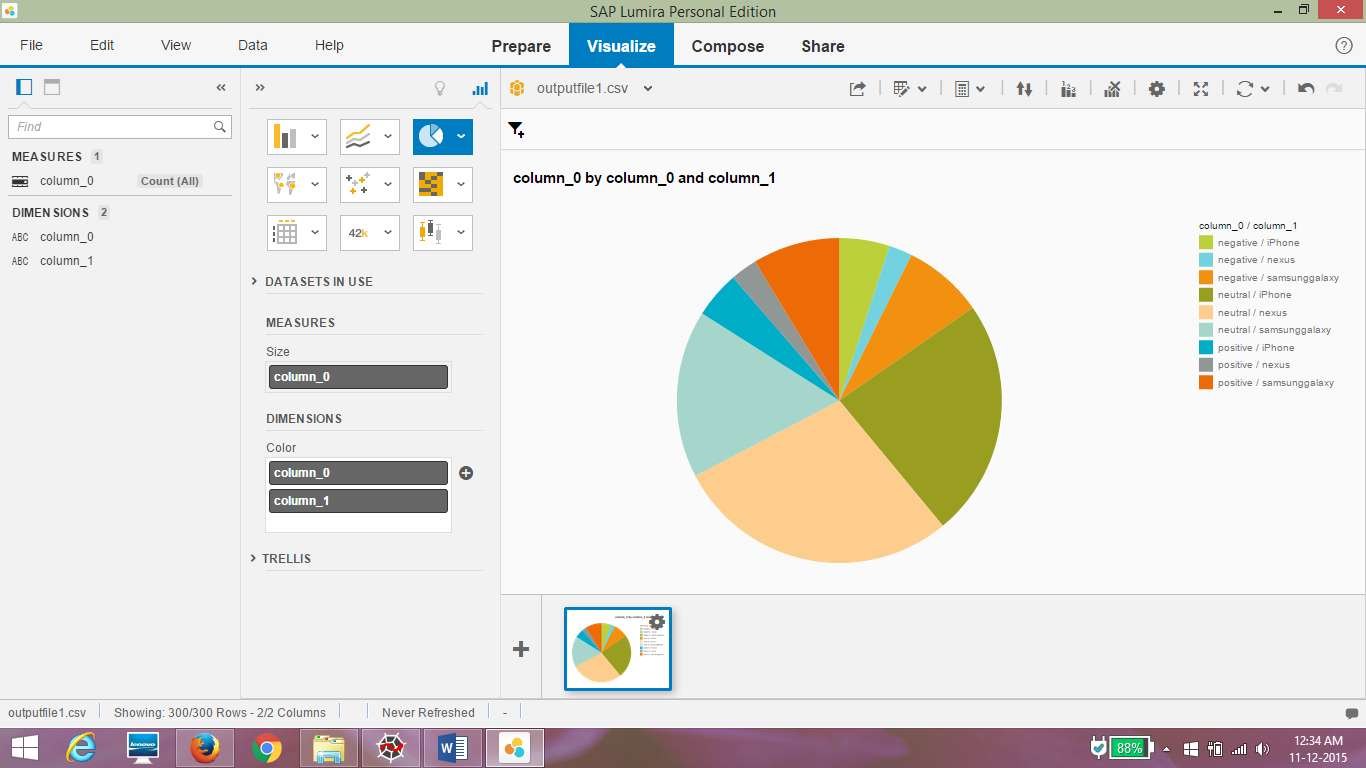


Naïve Bayes Classifier present in NLTK



SAP Lumira visualization tools which displays a graphical representation of the results.





**8. CONCLUSION AND SCOPE FOR FUTURE WORK**

**SCOPE FOR FUTURE WORK**

The overall efficiency of the twitter sentiment analyzer can be improved.

The following can be incorporated:

**Semantics:** The algorithms classify the overall sentiment of a tweet. The polarity of a tweet may depend on the perspective you are interpreting the tweet from. For example, in the tweet “India beat Australia :)”, the sentiment is positive for India and negative for Australia. In this case, semantics may help. Using a semantic role labeler may indicate which noun is mainly associated with the verb and the classification would take place accordingly. This may allow “India beat Australia :)” to be classified differently from “Australia beat India :)”.

**Bigger Dataset:** The training dataset in the order of millions will cover a better range of twitter words and hence better unigram feature vector resulting in an overall improved model. This would vastly improve upon the existing classifier results.

**Internationalization:** Currently, we have focused only on English tweets but Twitter has a huge international audience. It should be possible to use this approach to classify sentiment in other languages with a language specific positive/negative keyword list.

**Interpreting Sarcasm:** The present approach is currently incapable of interpreting sarcasm. In general sarcasm is the use of irony to mock or convey contempt, in the context of current work sarcasm transforms the polarity of an apparently positive or negative utterance into its opposite. This limitation can be overcome by exhaustive study of fundamentals in discourse-driven sentiment analysis".

The main goal of this approach is to empirically identify lexical and pragmatic factors that distinguish sarcastic, positive and negative

usage of words.

**CONCLUSION**

We presented results for sentiment analysis on Twitter to find the most preferred smartphone, by building a supervised system.

To conclude, this report has illustrated that an effective sentiment analysis can be by collecting a sample audience opinions from Twitter. Throughout the duration of this project many different data analysis tools were employed to collect, clean and mine sentiment from the dataset. Such an analysis could provide valuable feedback to customers who are would want to buy a new smartphone. Discovering negative trends early on can allow them to make informed choices. It is apparent from this study that the machine learning classifier used has a major effect on the overall accuracy of the analysis. Commonly used algorithm for text classification such as Naïve Bayes was examined. With machine learning algorithms constantly being developed and improved, massive amounts of computational power becoming readily available both locally and on the cloud, and unfathomable amounts of data being uploaded to social media sites every day, sentiment analysis will become standard practice for marketing and product feedback.

**9. REFERENCES**

1. Alec Go, Lei Huang, Richa Bhayani, 2009. Twitter Sentiment analysis, s.l.: The Stanford Natural Language Processing Group
2. B. Pang and L. Lee., “Opinion mining and sentiment analysis”, Foundations and Trends in Information Retrieval.
3. http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/